НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою
«Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему:
«*Система визначення параметрів автора за текстами творів*»

Виконав: студент *IV* курсу, групи *ІС-з61*
Мак Олексій Володимирович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *Доцент, к.т.н., доц. Фіногенов Олексій Дмитрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *Доцент, к.т.н., доц. Тєлишева Тамара Олексіївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *Доцент каф. ТК, ФІОТ Мелкумян Катерина Юріївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Маку Олексію Володимировичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Система визначення параметрів автора за текстами творів*»

керівник проєкту *Фіногенов Олексій Дмитрович, к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів
5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема структурна компонентів системи

3. Схема бази даних

4. Схема структурна класів програмного забезпечення

5. Схема структурна архітектури програмного забезпечення

6. Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*15.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*17.04.2020*	
3.	*Постановка та формалізація задачі*	*23.04.2020*	
4.	*Розробка інформаційного забезпечення*	*05.05.2020*	
5.	*Алгоритмізація задачі*	*01.05.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*05.05.2020*	
7.	*Розробка програмного забезпечення*	*19.05.2020*	
8.	*Налагодження програми*	*24.05.2020*	
9.	*Виконання графічних документів*	*25.05.2020*	
10.	*Оформлення пояснювальної записки*	*25.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Олексій МАК

Керівник Олексій ФІНОГЕНОВ

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6111.00.000 ПЗ	Пояснювальна записка	99	
3	А4	ДП 6111.01.000 ТЗ	Технічне завдання	9	
4	А3	ДП 6111.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6111.03.000 ССА	Схема структурна архітектури	1	
			програмного забезпечення		
6	А3	ДП 6111.04.000 СБД	Схема бази даних	1	
7	А3	ДП 6111.05.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
9	А3		Рішення з математичного забезпечення	1	
10	А3	ДП 6111.06.000 КЕ	Креслення вигляду екранних форм	3	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6111 00.000.00		
Розробн.		Мак О.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Фіногенов О.Д.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-з61	
Н/контр.		Тєлишева Т.О.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: *«Система визначення параметрів автора за текстами творів»*

Київ – 2020 року

Пояснювальна записка
до дипломного проєкту

на тему: *«Система визначення параметрів автора за текстами творів»*

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з шести розділів, містить 22 рисунки, 11 таблиць, 1 додаток, 19 джерел.

Дипломний проєкт присвячений вирішенню комплексу задач зі збору, обробки, параметризації та атрибуції авторства текстів українською мовою. Метою роботи є збір корпусу текстів літературних творів українських письменників та проведення досліджень щодо можливості атрибуції авторства текстів за допомогою статистичного аналізу, на основі буквосполучень різних порядків – n-грамів.

У розділі загальних положень описано етапи створення системи статистичного аналізу текстів за допомогою n-грамів, наведено функціональну модель системи, зазначено її відмінності від аналогів.

У розділі інформаційного забезпечення описано формат вхідних та вихідних даних, структури бази даних та файлового сховища для збереження результатів роботи програмного продукту.

Розділ математичного забезпечення присвячений формалізації обраного методу статистичного аналізу текстової інформації та обґрунтуванню доцільності його використання.

Розділ програмного забезпечення описує архітектуру, етапи проектування, методологію та засоби розробки програмного продукту; містить діаграму класів та специфікацію функцій.

У технологічному розділі вказано мету проведення випробувань програмного продукту та описано отримані результати.

ОБРОБКА ТЕКСТІВ, АТРИБУЦІЯ АВТОРСТВА, КОРПУСНА ЛІНГВІСТИКА, УКРАЇНСЬКА МОВА, N-ГРАМ.

ABSTRACT

The structure and the scope of the work. Explanatory note of the diploma project consists of six sections, contains 22 drawings, 11 tables, 1 application and 19 sources.

This diploma project is devoted to solving a set of problems in the collection, processing, defining parameters, and attributing authorship of texts in the Ukrainian language. The aim of the work is to collect the body of literary works of Ukrainian writers and conduct research on the possibility of attribution of authorship of texts through statistical analysis, based on letter combinations of different orders – n-grams.

The information section describes the format of input and output data, the structure of the database and file storage to save the results of the software product.

The section of mathematical support is devoted to the formalization of the chosen method of statistical analysis of textual information and substantiation of the expediency of its use.

The software section describes the architecture, design stages, methodology, and software development tools; it contains a class diagram and a specification of the functions.

The technological section indicates the purpose of testing the software product and describes the results.

TEXT PROCESSING, AUTHORSHIP ATTRIBUTION, CORPUS LINGUISTICS, UKRAINIAN LANGUAGE, N-GRAM.

ЗМІСТ

ВСТУП **4**

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ **6**

1.1 Опис предметного середовища 6

1.2 Опис процесу діяльності *8*

1.3 Опис функціональної моделі *10*

1.4 Огляд наявних аналогів *15*

1.5 Постановка задачі *18*

1.6 Призначення розробки *18*

1.7 Цілі та задачі розробки *18*

Висновок до розділу *18*

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ **20**

2.1 Вхідні дані *20*

2.2 Вихідні дані *21*

2.3 Опис структури бази даних *23*

2.4 Структура масивів інформації *27*

Висновок до розділу *27*

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ **29**

3.1 Змістовна постановка задачі *29*

3.2 Математична постановка задачі *29*

3.3 Обґрунтування методу розв'язання 30

3.4 Опис методів розв'язання *32*

Висновок до розділу *33*

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ **35**

4.1 Засоби розробки *35*

4.2 Вимоги до технічного забезпечення 37

4.3 Загальні вимоги *38*

4.4 Архітектура програмного забезпечення *38*

4.5 Діаграма класів *39*

4.6 Діаграма послідовності *40*

4.7 Діаграма компонентів *41*

4.8 Специфікація функцій *42*

Висновок до розділу 44

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ **46**

5.1 Керівництво користувача 46

5.2 Випробування програмного продукту *52*

5.3 Мета випробувань *53*

5.4 Загальні положення *53*

5.5 Результати випробувань *53*

Висновок до розділу *57*

ЗАГАЛЬНІ ВИСНОВКИ **59**

ПЕРЕЛІК ПОСИЛАНЬ **61**

ДОДАТОК А **63**

ВСТУП

Визначення авторства тексту вже багато років є актуальною задачею як в теоретичній лінгвістиці, так і в галузі прикладних задач. Атрибуція авторства застосовується у філологічних дисциплінах, в психології та теорії штучного інтелекту, в історичній та культурологічній сферах. Алгоритми визначення та порівняння авторської стилеметрії активно застосовуються у пошукових мережах, які вже давно стали невід'ємною частиною повсякденного життя пересічної людини. Завдяки масовому переходу від аналогових інформаційних носіїв до електронних, а також стрімке зростання обчислювальних потужностей, виникає безліч можливостей для впровадження систем комп'ютеризованого визначення авторства.

Незважаючи на те, що створення математичних методів, націлених на дослідження текстової інформації триває багато десятиліть, проте, поки що, не створено формалізованого методу визначення авторства тексту, який давав би гарантований результат. Слід зазначити, що завдання ускладнює велика кількість природніх мов, адже при дослідженні необхідно враховувати лексичні, семантичні та синтаксичні особливості, присутні в кожній з них.

Для нашої країни цей аспект набуває особливого значення, оскільки процес формування української мови, протягом усієї історії, відбувається в умовах стрімких та динамічних змін. Чисельні зміни геополітичної орієнтації у різних районах країни сприяють створенню унікальних культурологічних кліматів, які впливають на діалектичні особливості мови.

Нажаль, дослідженню літературного надбання не приділяється достатньо уваги, хоча саме корпус текстів літературних творів є одним з найбільш приємних об'єктів дослідження і може з великою ефективністю використовуватись для вивчення комп'ютерної лінгвістики. Саме тому, задача створення корпусу літературних творів українських авторів та подальша їхня обробка з метою параметризації, класифікації та атрибуції авторства є привабливою задачею дипломного проєкту.

Практичне значення одержаних результатів.

Розроблено набір інструментів для підготовки текстів, написаних українською мовою, для проведення лінгвістичних досліджень.

Проведено збір, аналіз та попередню обробку літературних творів українських письменників, сформовано дата сети для проведення подальших досліджень.

Створено систему статистичного аналізу текстів українською мовою з метою визначення параметрів автора. Проведено дослідження роботи системи на основі сформованого дата сету.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Формалізація та вирішення задачі атрибуції авторства літературних творів знаходиться в сфері інтересів як літературознавців, так і математиків. З одного боку, ця задача має велику практичну цінність у багатьох суспільно-наукових сферах: історичній, правовій, культурологічній тощо, з іншого боку, викликає зацікавлення процес безпосереднього дослідження творчої роботи мозку з точки зору алгоритмічності цього процесу.

Проблема авторства як особливого роду творення певного артефакту, зокрема тексту – не нова, але в сучасному інформаційному просторі помітно коригується новими соціокультурними чинниками, масовим розповсюдженням соціальних мереж, відкритістю та легкодоступністю інформації найрізноманітнішого змісту та призначення, розмиванням меж між особистим і суспільним, появою нових видів текстотворчої діяльності, навіть нових професій, пов’язаних із створенням текстів. Особа як створювач тексту представлена на найрізноманітніших рівнях, у найрізноманітніших статусах та видах діяльності [1].

Для вирішення задачі встановлення авторства тексту неминуче доводиться звертатися до експертів. Експерти можуть ідентифікувати автора невідомого тексту або визначити належність твору іншому автору за допомогою характерних мовних особливостей і стилістичних прийомів. Безсумнівно, подібні дослідження трудомісткі, однак, оскільки завдання встановлення авторства текстів виникає в багатьох областях та має прикладний характер, постає питання про створення формальних методів її вирішення. В даний час, для атрибуції текстів, застосовуються підходи з теорії розпізнавання образів, математичної статистики та теорії ймовірностей, алгоритми нейронних мереж і кластерного аналізу та багато інших методів [2].

Атрибуція тексту – дослідження тексту з метою встановлення авторства або отримання будь-яких відомостей про автора й умови створення текстового

документа. Як зазначено в [3], завдання атрибуції можна розділити на ідентифікаційні та діагностичні.

Ідентифікаційні завдання дозволяють підтвердити або виключити авторство певної особи, перевірити той факт, що автором всього тексту була одна особа або ж, що особа, яка написала текст є при цьому його справжнім автором. Ідентифікаційні завдання вирішуються з припущення, що автор тексту відомий.

Діагностичні завдання дозволяють визначити особистісні характеристики автора (освітній рівень, рідна мова, знання іноземних мов, походження, місце постійного проживання тощо), а також визначити факт свідомого спотворення письмової мови. Діагностичні завдання вирішуються з припущення, що автор тексту невідомий. У цих випадках зазвичай неможливо зіставити досліджуваний текст з текстами автора.

Формальні методи зазвичай засновані на порівнянні обчислюваних стилеметричних характеристик текстів. У загальному випадку текст відображається у вектор обчислених для нього параметрів, кожен з яких об'єктивно характеризує певний набір особливостей тексту. Таким чином, текст графічно відображається в деяку точку n-мірного простору. При такій формалізацій авторський стиль також може бути представлений у вигляді аналогічного вектору параметрів – цим вектором буде вектор текстів, написаних даним автором [4].

Отже, задача атрибуції зводиться до обчислення певних числових характеристик тексту, та представлення його у вигляді вектору, або точки n-мірного простору. За умови, що ми маємо оцінку деяких текстів цього ж автора, достатньо оцінити відстань між цією точкою та іншими текстами. У цьому випадку відстань буде позначати відмінність між числовими характеристиками тексту, що аналізується, та тих, що вже було проаналізовано [4].

Використання таких методів дозволяє вирішувати обидві задачі атрибуції одними і тими ж методами, оскільки у пам’яті ЕОМ можна зберегти велику кількість проаналізованих текстів і аналізувати нові набагато швидше, ніж це може робити експерт.

Разом з тим, постають деякі труднощі. Першою проблемою є те, що для достатньої точності аналізу необхідна велика кількість текстів різних авторів, щоб характеристика їх стилю була якомога точнішою. Відповідно, якщо у базі авторів немає текстів автора, твір якого аналізується, дізнатись авторство буде неможливо.

Найбільшою ж проблемою постає вибір числових характеристик тексту. Однією з необхідних умов для вибору характеристики є її швидка обчислюваність. Для першої умови застосовують попередню обробку тексту, щоб прибрати компоненти, які не мають значення для аналізу. Ці методи можуть різнитися залежно від лексичних особливостей тієї чи іншої мови. Другою і основною умовою є те, щоб ця характеристика досить корректно характеризувала авторський стиль. Тобто, необхідно, щоб для різних авторів вона відрізнялась достатньо сильно, а також щоб її було важко свідомо контролювати. На сьогодні відомо багато характеристик, які застосовуються у формальних методах визначення авторства. Проте абсолютно точного формального методу аналізу тексту на авторства не існує [5].

Також перешкодою для алгоритмів визначення авторства може бути малий обсяг вхідного тексту [3, 5].

1.2 Опис процесу діяльності

Виконання проєкту з автоматизації визначення авторства літературних творів містить у собі декілька підготовчих етапів, пов’язаних зі збором та обробкою інформації для створення вхідного дата сету.

Етап збору інформації для створення корпусу літературних творів українських авторів є фундаментальною складовою усього проєкту. За

відсутності дата сету з текстами відомих авторів, ми не матимемо змоги визначити стилеметричні характеристики авторів, а отже й вирішити задачу атрибуції авторства тексту невідомого.

Для збору інформації було знайдено декілька онлайн бібліотек з літературними творами українських авторів [6], на яких, у відкритому доступі, міститься велика кількість творів українських авторів, а також їхні біографії, які були необхідні для визначення додаткових параметрів автора, таких як роки життя та регіони країни у яких автор мешкав протягом своєї творчої діяльності.

Для кожного з ресурсів було розроблено скрипти автоматичного парсингу, для знаходження усіх наявних на сайті творів та збереження їх із зазначенням авторства, з метою подальшої обробки. Скрипти було написано мовою Python, з використанням бібліотек Beautiful Soup та Requests. В залежності від ресурсу, тексти зберігались у форматі txt або html.

В результаті парсингу сайтів було зібрано більше 12 000 літературних творів українських авторів. Після збору текстів виконання проєкту перейшло до другого етапу – фільтрації та попередньої обробки отриманих творів.

Було проведено фільтрацію зібраних текстів. З файлів було вилучено інформацію, що не є частиною літературного твору безпосередньо, таку як коментарі, пояснення тощо. Після цього з вибірки було вилучено поетичні твори, а також усі тексти, що містили менше ніж 15 000 літер.

Також, було виконано задачу з видалення літер не кириличних символів (зазвичай латинських), використаних у написанні українських слів. Такі символи досить часто потрапляють до текстів під час оптичного розпізнавання. Вилучення таких символів є нетривіальною та трудомісткою задачею. Для її вирішення було написано набір функцій з використанням регулярних виразів.

В результаті виконання підготовчих етапів було створено корпус прозових літературних творів українських авторів для використання у системі. Детальна структура отриманого дата сету описана у розділі 2.1 Вхідні дані.

Наступним та основним етапом проєкту стала розробка системи для атрибуції авторства текстів на основі частотності появи буквосполучень у текстах творів. На відміну від досліджень описаних у [3, 7], було вирішено розробити систему, яка не буде обмежуватись аналізом n-грамів порядків від 1 до 3, а матиме можливість конфігурування, для створення та аналізу n-грамів вищих порядків.

Таким чином, за допомогою системи ми зможемо не лише перевірити результати досліджень для грамів, біграмів та триграмів, а й провести додаткові статистичні дослідження творів українською мовою для n-грамів вищих порядків.

1.3 Опис функціональної моделі

Система розрахована на використання двома дійовими особами – це користувач та адміністратор. Варіанти використання, що існують в системі для кожного з акторів, наведено у таблиці 1.

Таблиця 1.1 – Варіанти використання системи

Актор	**Варіант використання**
Користувач	Переглянути параметри бібліотеки
	Переглянути перелік авторів
	Переглянути параметри автора
	Переглянути тексти автора
	Проаналізувати текст
	Переглянути перелік звітів
	Переглянути звіт

Продовження таблиці 1.1

Актор	**Варіант використання**
Адміністратор	Додати авторів до бібліотеки
	Додати тексти авторів
	Додати інформацію про авторів
	Змінити налаштування системи

Відповідно до визначених варіантів використання, побудовано загальну модель варіантів використання, яка наведена на рисунку 1.1.

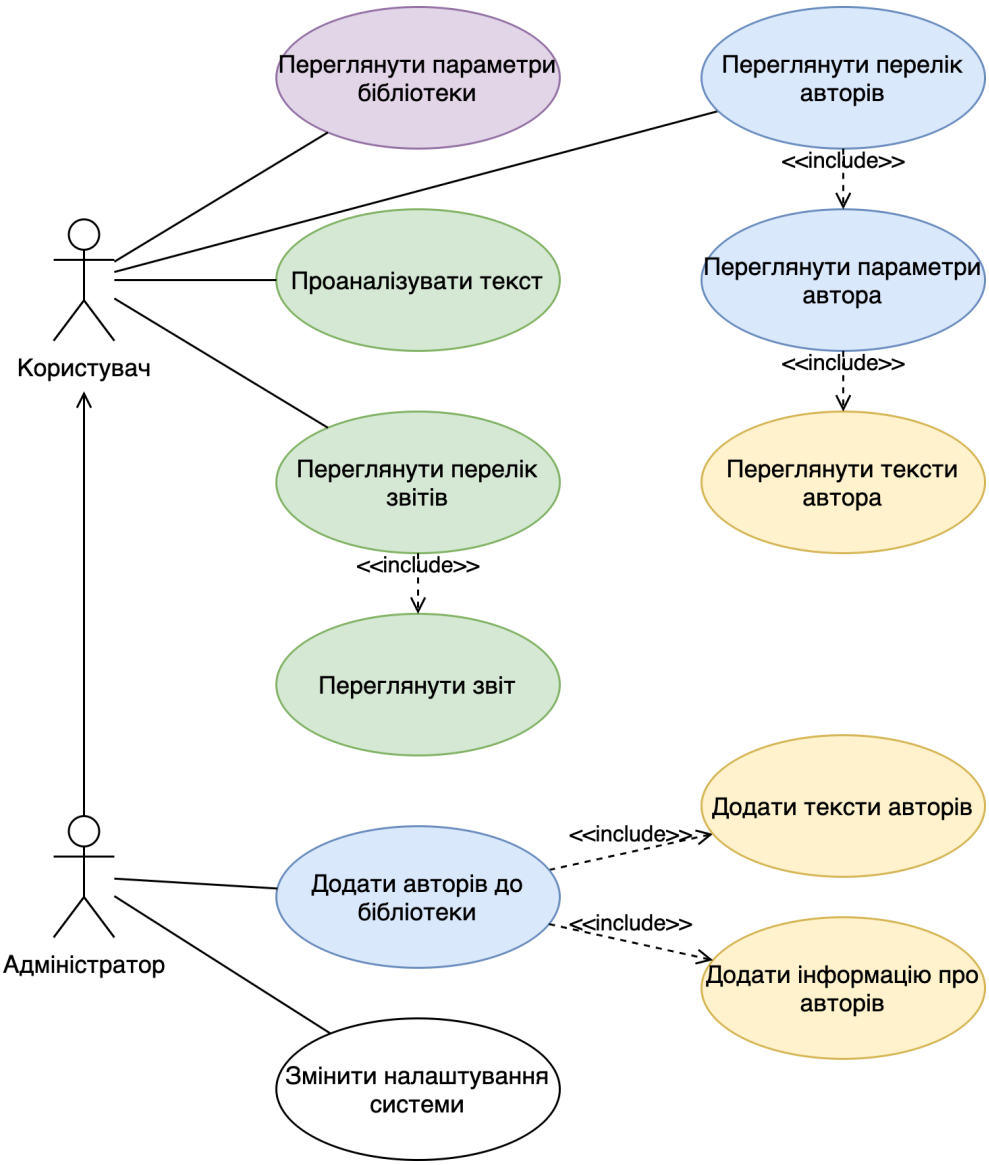


Рисунок 1.1 – Модель варіантів використання

У таблиці 1.2 наведено перелік функціональних вимог до варіантів використання.

Таблиця 1.2 – Вимоги з варіантів використання

Актор	**Варіант використання**	**Функціональна вимога**	**Пріоритет**
Користувач	Переглянути параметри бібліотеки	RQ 01. Система відображає сторінку з такими параметрами бібліотеки: − кількість авторів; − кількість текстів; − кількість слів; − кількість символів; − середня довжина слова; − слова, присутні у кожного з авторів; − діаграма еталонного біграму бібліотеки; − порівняльна діаграма частоти використання літер в бібліотеці та в українській мові в середньому.	Середній
Користувач	Переглянути перелік авторів	RQ 02. Система відображає сторінку з переліком авторів, для кожного автора зазначено параметри: − прізвище та ім’я; − кількість текстів; − кількість слів; − кількість символів; − середня довжина слова; − відстань до n-грамів бібліотеки.	Високий

Продовження таблиці 1.2

Актор	**Варіант використання**	**Функціональна вимога**	**Пріоритет**
Користувач	Переглянути параметри автора	RQ 03. Система відображає сторінку параметрами автора: − прізвище та ім’я; − порівняльна діаграма частоти використання літер автором та в українській мові в середньому; − діаграма еталонного біграму автора; − кількість текстів; − кількість слів; − кількість символів; − середня довжина слова; − відстань до n-грамів бібліотеки; − перелік текстів автора, для кожного з текстів зазначено параметри: назва, кількість слів, кількість символів, середня довжина слова, відстань до n-грамів автора; − найуживаніші слова.	Середній
Користувач	Переглянути тексти автора	RQ 04. Система відображає текст твору.	Низький

Продовження таблиці 1.2

Актор	**Варіант використання**	**Функціональна вимога**	**Пріоритет**
Користувач	Проаналізувати текст	RQ 05. Система надає можливість завантажити текстовий фрагмент для аналізу та вказати назву фрагменту. Після завантаження, відбувається аналіз тексту. Результати аналізу відображаються на сторінці звіту.	Високий
Користувач	Переглянути перелік звітів	RQ 06. Система відображає сторінку з переліком звітів, для кожного зі звітів зазначено параметри: − номер; − назва; − кількість слів; − кількість символів; − середня довжина слова.	Високий
Користувач	Переглянути звіт	RQ 07. Система відображає сторінку звіту з такими результатами аналізу тексту: − порівняльна діаграма частоти використання літер у тексті та в українській мові в середньому; − діаграма еталонного біграму тексту; − перелік найближчих до тексту авторів за кожним з n-грамів; − текст, який аналізувався.	Високий

Продовження таблиці 1.2

Актор	Варіант використання	Функціональна вимога	Пріоритет
Адміністратор	Додати авторів до бібліотеки	RQ 08. Система надає можливість завантажити тексти та інформацію про авторів у файлове сховище.	Високий
Адміністратор	Змінити налаштування системи	RQ 09. Система надає можливість змінити такі налаштування системи: − порядок n-грамів; − розміщення файлового сховища; − параметри підключення до бази даних; − кількість потоків паралельного обчислення.	Середній

1.4 Огляд наявних аналогів

Нажаль, відомих відкритих систем, призначених для дослідження та атрибуції авторства текстів, українською мовою, немає. Тому наведемо приклади програмного забезпечення, призначеного для визначення авторства тексту російською мовою. Розглянемо використані в них методи аналізу та опишемо дослідження, які проводились з використанням цих систем, наведені у [2].

Система «Лінгвоаналізатор»

Система «Лінгвоаналізатор» демонструє можливості математичного аналізу структури тексту. Метою аналізу є визначення близькості тексту до одного з авторських еталонів, визначених заздалегідь [7].

Програма аналізує вхідний текст і видає імена трьох письменників, які могли б бути його найбільш ймовірними авторами. Крім імен письменників,

програма знаходить три твори кожного з авторів, найбільш близькі до даного тексту.

Стилеметричну функціональність системи реалізовано з використанням двобуквених буквосполучень – біграмів.

Система «Атрибутор»

Для визначення стилістичної характеристики тексту, система використовує n-грами. Крім біграмів – двобуквених буквосполучень, у системі використовуються триграми – трибуквені буквосполучення. Завдяки цьому, аналізу піддаються однобуквені та двобуквені службові частини мови, такі як прийменник, частка, сполучник, які є важливими стилістичними параметрами для визначення авторства [8].

В результаті дослідження було з'ясовано, що саме триграми є найбільш ефективною стилеметричною характеристикою тексту, у порівнянні з біграмами, або n-грамами вищих порядків.

У базі даних системи зберігаються романи 103 відомих російських авторів. Вибірку було підібрано таким чином, щоб тексти різних письменників максимально відрізнялись один від одного, а тексти одного письменника були максимально подібними.

Для атрибуції тексту лінгвістичним процесором «Атрибутор», достатньо текстового фрагменту розміром 20Кб.

Система «СМАЛТ»

Система атрибуції текстів «СМАЛТ» (Статичні методи аналізу літературного тексту), заснована на алгоритмах автоматизації морфологічного та статистичного аналізу [9].

Робота системи відбувається у три етапи. На першому етапі відбувається розбиття вхідного тесту на лексичні одиниці: абзац, речення, слово. На другому, виконується безпосередній аналіз перетвореного тексту та

морфологічний розбір. На третьому етапі проводиться синтаксичний аналіз на основі морфологічного розбору.

Система «Автороведа»

Робота системи побудована з використанням нейронних мереж в поєднанні з методом опорних векторів. Визначення авторства відбувається за допомогою бінарного класифікатора. Всі тексти, включно з навчальною частиною вибірки, розгортаються в багатовимірний вектор, який індексується словами. Надалі розглядається дві множини точок з навчальної вибірки в багатовимірному просторі. Множина точок, що належать даному автору та множина, яка не належить автору. Для відокремлення цих множин, простір поділяється на дві частини, за допомогою побудованої гіперплощини. У системі «Автороведа», гіперплощина будується за допомогою методу опорних векторів (SVM – Support Vector Machines). Після цього для класифікації тексту з невідомим автором перевіряється, в яку частину простору він потрапив [3].

В якості стилеметричної характеристики тексту система використовує найчастіші триграми символів та найуживані слова російської мови.

Для атрибуції текстів у системі «Автороведа» використовуються штучні нейронні мережі архітектури багатошаровий перцептрон (MLP), мережі каскадної кореляції (CCN), а також, апарат машини опорних векторів (SVM). CCN дозволяють знизити часові витрати на навчання, в порівнянні з перцептроном, за рахунок алгоритму автоматичної побудови топології мережі. SVM є найточнішим та найшвидшим з існуючих методів класифікації.

Дослідження проводились на текстовому корпусі, що складається з 215 прозових текстів 50 російських письменників. Розмір кожного тексту становив понад 100000 символів. Використовувалися вибірки об'ємом 1000–100000 символів (200–20000 слів). Було створено по 3 навчальних вибірки кожного автора, для тестування використовувалося по 1 вибірці автора. Дослідження показали, що найінформативнішими авторським ознаками є 300–700 найбільш

частотних триграми і 500 найчастіших слів. Точність визначення автора становить 0,95–0,98 для текстового фрагменту 20000–25000 символів.

1.5 Постановка задачі

1.6 Призначення розробки

Розробка призначена для вирішення таких задач: збір корпусу текстів літературних творів українських письменників, попередня обробка зібраних текстів та стилеметрична параметризаця текстів з метою класифікації та атрибуції авторства.

1.7 Цілі та задачі розробки

Основною метою розробки є створення системи атрибуції авторства текстів та збір корпусу літературних творів українських письменників для проведення досліджень з визначення авторства невідомого твору за допомогою створеної системи.

Створення масштабного дата сету дасть змогу обчислити еталонні стилеметричні показники для великої кількості авторів, присутніх у ньому.

Таким чином, за допомогою цієї розробки буде можливо не лише перевірити математичну модель, обрану для вирішення задачі атрибуції текстів, а й проводити більш масштабні дослідження, завдяки наявності у зібраному корпусі текстів великої кількості авторів.

Висновок до розділу

У першому розділі описано предметну область та проаналізовано проблему атрибуції авторства літературних творів українських письменників. Перелічено попередні етапи зі збору, фільтрації та обробки текстів з метою створення корпусу літературних творів для проведення подальших

досліджень. В розділі сформульовано постановку задачі та призначення системи, описана функціональна модель та визначені дійові особи.

Наведено діаграму варіантів використання, що відображає функції, які може виконувати система визначення автора тексту та перелік функціональних складових. Також наведена таблиця з функціональними вимогами щодо роботи застосунку за кожним варіантом використання.

В розділі перелічено наявні аналоги, проведено аналіз їх функціональних особливостей та описано математичні методи, які застосовуються у їхній роботі.

Крім призначення системи, у розділі описано цілі, поставлені перед розроблюваною системою та перелічено задачі які можуть бути вирішені завдяки виконанню цього проєкту.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Для виконання автоматизованого аналізу даних, система використовує дата сет, який являє собою ієрархічну структуру папок з текстовими та конфігураційними файлами.

Дата сет міститься у папці uploads, розташування якої може бути налаштоване адміністратором системи. На рисунку 2.1 наведено структуру вхідного дата сету.

```
└── uploads
    ├── Прізвище-автора-1 Ім'я-автора-1
    │   ├── info.yml
    │   ├── Назва твору 1.txt
    │   ├── Назва твору 2.txt
    │   ├── Назва твору n.txt
    └── Прізвище-автора-n Ім'я-автора-n
        ├── info.yml
        ├── Назва твору 1.txt
        ├── Назва твору 2.txt
        └── Назва твору n.txt
```

Рисунок 2.1 – Структура вхідного дата сету

Усі файли дата сету зберігаються у текстовому форматі з використанням кодування utf-8.

Файл info.yml є текстовим конфігураційним файлом, в якому міститься додаткова інформація про автора, а саме:

date_birth: дата народження;

date_death: дата смерті;

location: регіони, в яких проживав автор;

url_ukrlib: посилання на сторінку з біографією автора на сайті ukrlib.com.ua;

url_wiki: посилання на сторінку автора на сайті wikipedia.org.

Завдяки текстовому формату, файл зручно редагувати у будь-якому текстовому редакторі.

Дата сет, що використовувався для розробки, тестування та проведення досліджень з використанням розробленої аналітичної системи було створено на етапах збору та попередньої обробки інформації.

2.2 Вихідні дані

Користувач взаємодіє з системою за допомогою вебінтерфейсу, результатом такої взаємодії є вебсторінки, які містять усі вихідні дані з параметрами бібліотеки, параметрами авторів та їхніх текстів (рисунок 2.2), а також результати звітів про проведення аналізу текстових фрагментів (рисунок 2.3).

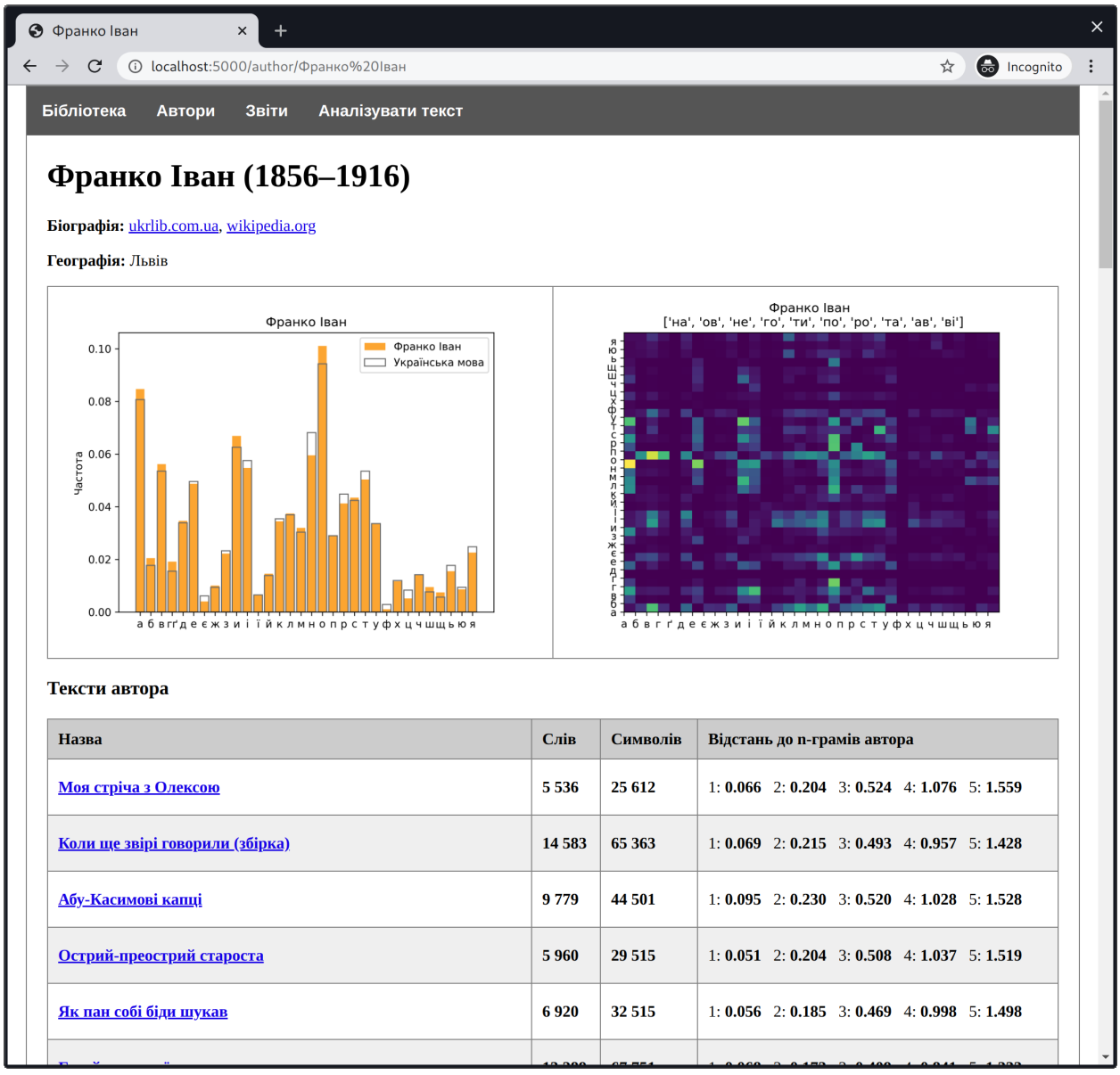


Рисунок 2.2 – Сторінка параметрів автора

Рисунок 2.3 – Сторінка звіту

На етапі додавання авторів до системи, а також під час проведення аналізу текстових фрагментів, системою генеруються діаграми грамів та біграмів, які відображаються на відповідних вебсторінках.

Графічні файли діаграм авторів бібліотеки зберігаються у форматі png. Тексти творів зберігаються у текстовому форматі з використанням кодування utf-8. Діаграми та тексти розміщуються в папці authors, розташування якої може змінювати адміністратор системи. Папка authors має структуру, наведену на рисунку 2.4.

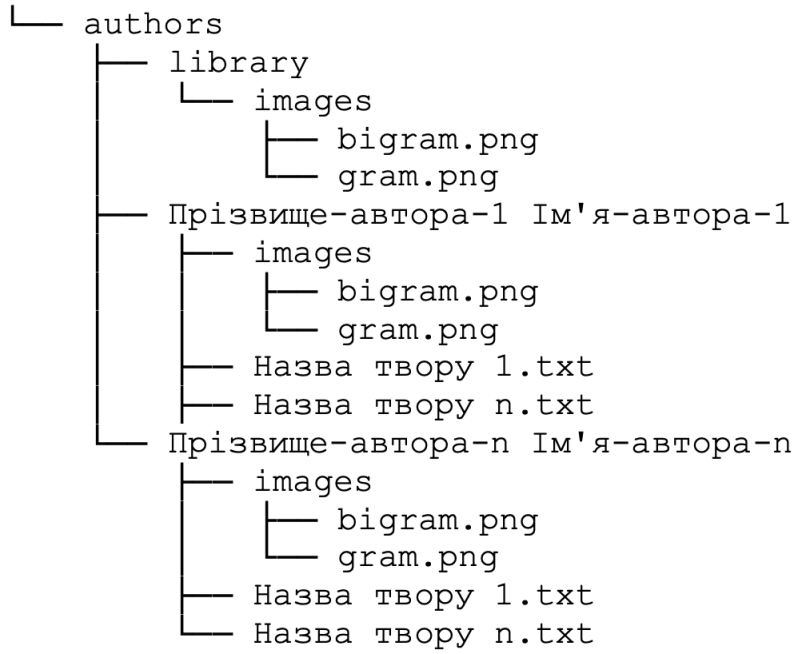


Рисунок 2.4 – Структура сховища діаграм та текстів авторів

Папка з діаграмами звітів містить діаграми грамів та біграмів, і має структуру наведену на рисунку 2.5.

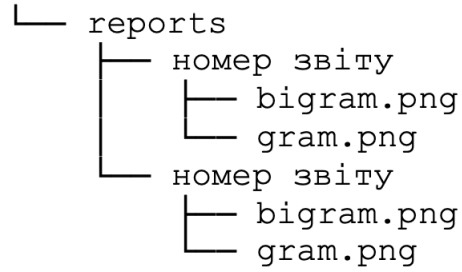


Рисунок 2.5 – Структура сховища діаграм звітів

2.3 Опис структури бази даних

Для збереження даних у системі використовується СКБД PostgreSQL. У базі даних зберігається інформація про параметри бібліотеки, параметри авторів та їхніх творів, а також інформація про звіти, отримані в результаті аналізу текстових фрагментів.

База даних складається з чотирьох таблиць, дві з яких пов'язані відношенням один до багатьох.

На рисунку 2.6 наведено ER діаграму бази даних, на якій зазначено назви таблиць, назви та типи полів, а також типи зв’язків між таблицями.

Рисунок 2.6 – ER діаграма бази даних

У таблицях 2.1–2.4 деталізовано типи та призначення полів для кожної з таблиць бази даних.

Таблиця 2.1 – Детальний опис таблиці library

Поле	Тип	Призначення
id	integer PK	Унікальний ідентифікатор запису в таблиці (є первинним ключем)
title	varchar(256)	Назва бібліотеки (корпусу текстів)
n-authors	integer	Кількість авторів у бібліотеці
n-texts	integer	Кількість текстів у бібліотеці
n-words	integer	Кількість слів у бібліотеці
n-letters	integer	Кількість літер у бібліотеці
common_words	bytea	Словник спільних слів для всіх авторів у бібліотеці

Таблиця 2.2 – Детальний опис таблиці reports

Поле	Тип	Призначення
id	integer PK	Унікальний ідентифікатор запису в таблиці (є первинним ключем)
title	varchar(256)	Назва тексту для якого проводився аналіз
text	text	Текст, аналіз якого проводився
n-words	integer	Кількість слів у бібліотеці
n-letters	integer	Кількість літер у бібліотеці
distance_to_authors	bytea	Список відстаней до найближчих авторів для кожного з n-грамів

Таблиця 2.3 – Детальний опис таблиці authors

Поле	Тип	Призначення
id	integer PK	Унікальний ідентифікатор запису в таблиці (є первинним ключем)
first_name	varchar(64)	Ім’я автора

Продовження таблиці 2.3

Поле	Тип	Призначення
last_name	varchar(64)	Прізвище автора
date_birth	date	Рік народження автора
date_death	date	Рік смерті автора
n-texts	integer	Кількість текстів
n-words	integer	Кількість слів
n-letters	integer	Кількість літер
distance_to_library	bytea	Список відстаней до еталонних n-грамів бібліотеки
vocabulary	bytea	Словник 50 найуживаніших автором слів за виключенням слів, спільних для усіх авторів
location	varchar(128)	Регіони, у яких працював автор
url_ukrlib	varchar(128)	URL адреса сторінки з біографією автора на сайті ukrlib.com.ua
url_wiki	varchar(512)	URL адреса сторінки автора на сайті wikipedia.org

Таблиця 2.4 – Детальний опис таблиці texts

Поле	Тип	Призначення
id	integer PK	Унікальний ідентифікатор запису в таблиці (є первинним ключем)
author_id	integer FK	Номер унікального ідентифікатора автора тексту (є зовнішнім ключем)
title	varchar(256)	Назва тексту
text	text	Текст
n-words	integer	Кількість слів у тексті
n-letters	integer	Кількість літер у тексті
distance_to_author	bytea	Список відстаней до еталонних n-грамів автора

2.4 Структура масивів інформації

Для проведення досліджень з використанням високих порядків n-грамів та великої кількості авторів, паралельно з реляційною базою даних, система використовує файлове сховище для збереження серіалізованих об’єктів у бінарному форматі. Така архітектура дозволяє працювати з об’єктами великого об’єму, не створюючи велике навантаження на базу даних та не сповільнюючи роботу вебінтерфейсу.

Сховище бінарних об’єктів зберігається у папці blobs, розміщення якої може змінювати адміністратор системи, та має структуру, зображену на рисунку.

```
└── blobs
    ├── library
    │   ├── library.pickle
    ├── Прізвище-автора-2 Ім'я-автора-2
    │   ├── author.pickle
    ├── Прізвище-автора-2 Ім'я-автора-2
    │   ├── author.pickle
    └── Прізвище-автора-n Ім'я-автора-n
        └── author.pickle
```

Рисунок 2.7 – Структура файлового сховища бінарних об’єктів

Висновок до розділу

Розділ містить вичерпну інформацію щодо форматів вхідних та вихідних даних, які використовуються в роботі системи атрибуції авторства текстів.

У підрозділі вхідних даних описано структуру дата сету, створеного на етапі збору та попередньої обробки інформації, який використовується для роботи з системою та проведення аналітичних досліджень. Зазначено формати та кодування файлів вхідного дата сету.

Підрозділ вихідних даних містить інформацію про структуру сховищ та формати вихідних файлів, а також приклади екранних форм з результатами роботи системи.

У наступному підрозділі описано структуру реляційної бази даних та наведено ER діаграму з описом таблиць та зв’язків між ними.

Останній підрозділ містить інформацію про сховище серіалізованих бінарних об’єктів, що створюються під час роботи системи визначення параметрів автора.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Нехай існує бібліотека, яка містить літературні твори певної кількості українських авторів. Маючи текст невідомого автора, написаний українською мовою, необхідно визначити найбільш імовірного автора з представлених у бібліотеці.

Оскільки задача визначення авторства немає чіткого формалізованого рішення, необхідно використати метод статистичного аналізу, вважаючи основним стилеметричним параметром тексту набір n-грамів.

Отже, постає задача знаходження набору з n авторів, для яких відстань між n-грамом тексту та еталонним n-грамом автора буде найменшою.

Описаний підхід може бути застосований і до завдань ідентифікації текстів за іншими параметрами, якими можна характеризувати твори, крім авторської приналежності: роки написання, географічне розташування, тематичне спрямування, літературний жанр тощо. Для завдання ідентифікації автора підхід порівняння з еталоном є достатньо ефективним, для використання у прикладній сфері.

Це пов'язано з тим, що ширина розкиду відстаней між текстами одного автора дещо більше, ніж ширина розкиду відстаней між ними ж і середньої авторської частотності n-грамів. Для жанрів метод середнього «жанрового зразка» може бути не особливо ефективний, тому що при великій кількості авторів, які беруть участь у формуванні такого еталона, частотність n-грамів буде наближатися до деякої типової норми для всієї мови загалом, і точність такого інструменту виявиться низькою [4].

3.2 Математична постановка задачі

Нехай існує бібліотека, яка містить тексти A авторів. Дано текст Z, невідомого автора. Необхідно побудувати вектори щільності розподілу

буквосполучень (n-грамів) для порядків від 1 до n (n задається у налаштуваннях системи), та для кожного з порядків n, щільності функції розподілу n-грамів, визначити автора, для якого відстань ρ_{az}^{n} між вектором n-граму тексту та еталонним вектором n-граму є мінімальною.

3.3 Обґрунтування методу розв'язання

Оскільки кількість різних букв у алфавіті кожної мови обмежена, важливою характеристикою тексту є повторюваність буквосполучень: біграмів, триграмів та n-грамів, у загальному випадку. Важливою особливістю цих стилеметричних характеристик є те, що вони мають досить високу стійкість.

У таблиці 3.1 наведено частоти використання букв для сучасної української мови, обчислені на основі аналізу текстів художнього, публіцистичного та технічного спрямування [10].

Таблиця 3.1 – Частоти використання букв української мови

Буква	Частота	Буква	Частота
А	0.0807	Н	0.0681
Б	0.0177	О	0.0942
В	0.0535	П	0.0290
Г, Ґ	0.0155	Р	0.0448
Д	0.0338	С	0.0424
Е	0.0495	Т	0.0535
Є	0.0061	У	0.0336
Ж	0.0093	Ф	0.0028
З	0.0232	Х	0.0119
И	0.0626	Ц	0.0083

Продовження таблиці 3.1

Буква	Частота	Буква	Частота
І	0.0575	Ч	0.0141
Ї	0.0065	Ш	0.0076
Й	0.0138	Щ	0.0056
К	0.0354	Ь	0.0177
Л	0.0369	Ю	0.0093
М	0.0303	Я	0.0248

Для кращої наочності наведеної у таблиці інформації, побудуємо гістограму частотності букв у сучасній українській мові. З наведеної на рисунку 3.1 гістограми чудово видно різницю у частотності використання літер.

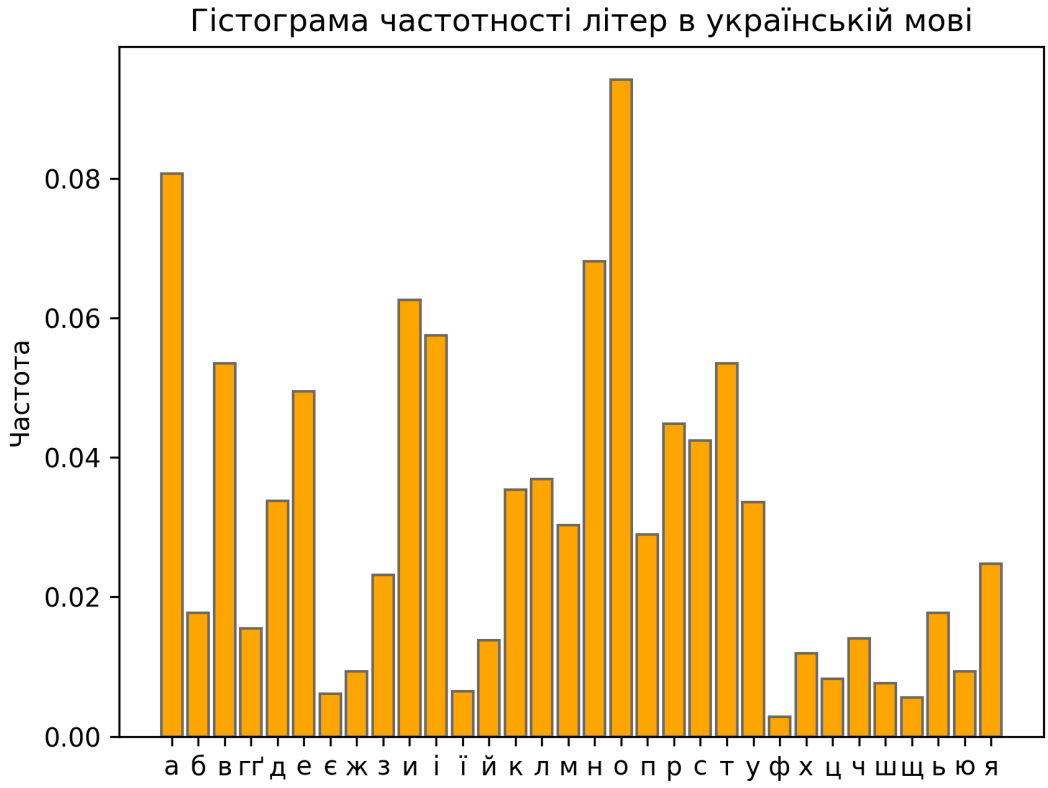


Рисунок 3.1 – Гістограма частотності використання букв в українській мові.

Якщо $\vartheta(g_{i1}, g_{i1}, \dots, g_{in})$ – кількість появ n-грама $g_{i1}, g_{i1}, \dots, g_{in}$ у тексті T, а L – загальна кількість n-грамів, то при досить великих L, частоти

$$\frac{\vartheta(g_{i1}, g_{i1}, \dots, g_{in})}{L} \quad (3.1)$$

для даного n-граму мало відрізняються одна від одної. Виходячи з цього, відносну частоту (3.1) вважають наближеною ймовірністю $P(g_{i1}, g_{i1}, \dots, g_{in})$ появи даного n-граму у випадково обраному місці тексту (за статистичним визначенням ймовірності) [12].

Частотний аналіз використання n-грамів показав що частота використання букв та буквосполучень відрізняється як для різних мов, що використовують один алфавіт [11], так і для текстів різної специфіки всередині однієї мови [13, 14]. Дослідження літературних творів показали що частота використання n-грамів відрізняється для різних авторів, отже може бути використана для вирішення задачі атрибуції авторства [4].

3.4 Опис методів розв'язання

Формалізуємо задачу ідентифікації автора невідомого тексту. Припустимо що тексти у бібліотеці, представлені у вигляді щільності функції розподілу n-грамів для A відомих авторів. Позначимо кількість текстів a–го автора через K_a, кількість літер в i-му тексті цього автора через $N_{i,a}$, де $i = 1, 2, \dots, K_a$. Вважатимемо, що довжина кожного з текстів достатня для проведення статистичного аналізу [4].

Через $f_{i,a}^{n}(j)$ позначимо щільність функції розподілу n-грамів відповідного тексту, де аргумент $j = 1, \dots, a(n) = 1, \dots, 33^n$. Для кожного автора визначимо середньозважене значення щільності функції розподілу:

$$F_a^n(j) = \frac{1}{N_a} \sum_{i=1}^{K_a} f_{i,a}^n(\mathrm{j}) N_{i,a}, \qquad N_a = \sum_{i=0}^{K_a} N_{i,a} \qquad (3.2)$$

Значення $F_a^n(j)$ вважатимемо авторським еталоном для кожного з n-грамів.

Позначимо через ρ_{ik}^n відстань між щільністю функцій розподілу двох текстів i та k:

$$\rho_{ik}^n = \|f_i^n - f_k^n\| = \sum_{j=1}^{a(n)} |f_i^n(j) - f_k^n(j)| \qquad (3.3)$$

Нехай є текст Z невідомого автора. Для кожного з n-грамів знайдемо автора, для якого відстань $\rho_{az}^n = \|f_z^n - F_a^n\|$ між щільністю функції розподілу $f_z^n(j)$ тексту Z та еталонною авторською щільністю функції розподілу $F_a^n(j)$ мінімальна:

$$\rho_{az}^n = \|f_z^n - F_a^n\| \qquad (3.4)$$

Таким чином отримаємо найбільш імовірних авторів невідомого тексту для кожного з n-грамів.

Висновок до розділу

Під час написання даного розділу дипломного проєкту було сформульовано змістовну та математичну постановки задачі, визначено основні терміни, які будуть використовуватися.

В змістовній постановці задачі описано, в загальному вигляді, задачу атрибуції авторства за допомогою статистичного аналізу, а саме за допомогою n-грамів.

В математичні постановці задачі було формалізовано саму задачу та наведено її основні атрибути. Визначено математичний запис задачі.

Також, було наведено опис методів розв’язання задачі. Окрім цього було наведено математичне обґрунтування обраного методу.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Вибір мови програмування для вирішення поставленої задачі, враховуючи предметну область та специфіку усіх етапів розробки системи, не виявився складним. Найпопулярнішою мовою для вирішення задач машинного навчання та аналітики даних сьогодні є Python. Окрім наявності справді великої кількості бібліотек для обробки даних, мова чудово допомагає вирішити й усі інші завдання, пов’язані з розробкою системи, такі як парсинг вебсторінок, робота з базою даних з використанням ORM системи, робота з файловою системою та великими масивами бінарних даних, розробка систем з клієнт-серверною архітектурою та паралельні обчислення.

Розробку мови розпочав у 1989 голландський вчений Гвідо ван Россумом, а вже у 1991 році світ побачила перша публічна версія цієї чудової мови програмування. З того часу, популярність мови невпинно зростає, а за останні кілька років, завдяки своїй зручності для використання у сфері аналізу даних, вона стала найбільш використовуваною мовою програмування у світі.

Python – це легка у вивченні, потужна мова програмування. Вона має ефективні структури даних високого рівня та просту, але ефективну реалізацію об'єктно-орієнтованої парадигми програмування. Елегантний синтаксис та динамічна типізація Python разом із інтерпретованою суттю роблять його ідеальною мовою для програмування та швидкого розвитку додатків у багатьох областях на більшості платформ [15].

Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна та

функціональна. Основною перевагою мови Python є велика кількість різноманітних бібліотек.

Мова Python є потужним інструментом для розробки серверної частини вебдодатків. Завдяки таким фреймворкам, як Django та Flask можна розробляти як монолітні додатки, так і додатки, що базуються на мікросервісній архітектурі. Існуючі бібліотеки дають можливість обирати між вже визначеною структурою системи, яка забезпечує легкість розробки і стабільність в роботі, та можливістю формувати структуру розробнику, що надає системі гнучкість та легку масштабованість [16].

В якості вебфреймворка для побудови системи з вебінтерфейсом було обрано Flask. Цей мікрофреймворк не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм та інші високорівневі компоненти. В той же час існує велика кількість розширень для реалізації усієї необхідної функціональності. Завдяки такій мікроархітектурі, Flask дозволяє будувати додатки, що використовують лише ті компоненти, які необхідні для вирішення конкретної задачі. Таким чином вдається уникнути використання зайвих компонентів і досягнути високої швидкодії та чистоти програмного коду [17].

В якості системи керування базою даних було обрано PostgreSQL.

PostgreSQL – це потужна об'єктно-реляційна база даних із відкритим кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які гарантують цілісність даних та масштабованість у разі високого навантаження. Розробка PostgreSQL почалась у 1986 році в рамках проекту POSTGRES в Каліфорнійському університеті в Берклі. PostgreSQL заслужила чудову репутацію за свою перевірену архітектуру, надійність, цілісність даних, потужний набір функцій, розширюваність та прихильність концепції відкритого коду. PostgreSQL працює на всіх основних операційних системах та повністю сумісна з ACID [18].

Ця об'єктно-реляційна система керування базами даних з відкритим кодом повністю відповідає стандартам ANSI SQL-99 та задовольняє усі потреби, які постають перед СКБД у даному проєкті.

Для взаємодії розроблюваної системи з базою даних обрано (ORM – Object-relational mapping) систему SQLAlchemy, яка надає можливість створювати об’єктно-реляційні відображення для опису структури бази даних, а також доступу до цих даних без написання коду мовою SQL [19].

4.2 Вимоги до технічного забезпечення

Поставлена перед системою задача, а саме порівняння величезних векторів є справді дуже ресурсозатратною, з точки зору обчислювальних можливостей. Саме тому, для створення дата сету, обробки даних та проведення подальших досліджень було використано сервер, оснащений 8-ма ядрами та 16 гігабайтами оперативної пам’яті.

Хоча мінімальні вимоги до технічного забезпечення серверу для запуску системи є низькими:

- кількість ядер процесора – 1;
- об’єм оперативної пам’яті – 1 Гб;
- об’єм дискового накопичувача – 10 Гб;
- мережевий інтерфейс;
- операційна система сімейства Linux (CentOS, Debian, Ubuntu).

Для використання системи з великим дата сетом авторів та проведення масових аналізів невідомих текстів, система має задовільняти рекомендованій конфігурації:

- кількість ядер процесора – 8;
- об’єм оперативної пам’яті – 12 Гб;
- об’єм дискового накопичувача – 32 Гб;
- мережевий інтерфейс;
- операційна система сімейства Linux (CentOS, Debian, Ubuntu).

Незалежно від конфігурації, на сервері має бути встановлено середовище для виконання програм Python 3.8.

Вимоги до клієнтського технічного забезпечення є мінімальними, для роботи з системою потрібен будь-який сучасний веббраузер та мережеве підключення.

4.3 Загальні вимоги

Користувачі та адміністратор системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

4.4 Архітектура програмного забезпечення

Система має клієнт-серверну архітектуру. Клієнтська частина реалізована за допомогою вебінтерфейсу, з якою користувач взаємодіє за допомогою веббраузеру. Таким чином забезпечується можливість запуску системи як на локальній машині користувача, або на сервері в локальній мережі, так і на потужних хмарних серверах, для проведення досліджень з використанням дата сетів великих розмірів та для швидкого аналізу невідомих текстів.

У конфігурації, яка була створена для демонстрації системи, серверна частина складається з двох компонентів, вебсерверу та серверу баз даних. Між собою ці компоненти обмінюються даними через віртуальну локальну мережу. Зовнішній доступ можливий лише до вебсерверу за допомогою протоколу HTTP, для користувача та протоколу SSH для адміністратора. Завдяки тому, що доступ до серверу баз даних можливий лише через консоль вебсерверу, а доступ до вебсерверу обмежений лише двома портами, досягається високий рівень безпеки системи.

4.5 Діаграма класів

Основний компонент системи, відповідальний за завантаження даних авторів з файлового сховища, обчислення еталонних параметрів авторів та бібліотеки, а також за пошук найближчих авторів для невідомого тексту, реалізовано з використанням парадигми об’єктно-орієнтованого програмування.

Функціональність розподілена між чотирма класами: Gram, Text, Author, Library. Кожен з класів інкапсулює у собі функціональність для виконання покладених на нього задач. Класи взаємодіють між собою, створюючи лінійну ієрархічну структуру.

Клас Gram є базовим, він не містить у собі екземпляри інших класів та не містить функціональності для взаємодії з ними.

Клас Text містить у собі один екземпляр об’єкту класу грам Gram та використовує його для обчислення, збереження та проведення операцій над еталонними параметрами тексту.

Клас Author містить у собі колекцію екземплярів класу Text та додатково, один екземпляр класу Gram, для збереження еталонних параметрів.

Клас Library містить один екземпляр класу Gram, для збереження еталонних параметрів. Також клас Library оснащено функціональністю для взаємодії з класами Author та Text.

Докладна схема структури класів, із зазначенням параметрів та методів кожного класу, а також структурою міжкласової взаємодії зображена на рисунку 4.1.

Рисунок 4.1 – Діаграма класів

4.6 Діаграма послідовності

Розглянемо послідовність етапів підготовки та використання системи. Отже, першим є етап збору та попередньої обробки літературних творів. На цьому етапі відбувається парсинг сайтів онлайн бібліотек та збір наявних там текстів. Зібрані тексти піддаються фільтрації та попередній обробці, після чого з них формується базовий дата сет, який зберігається у файловому сховищі.

Другим етапом є завантаження та опрацювання дата сету для створення бібліотеки. На цьому етапі обраховуються еталонні n-грами та інші статистичні показники авторів, присутніх у дата сеті. Після цього створюється бібліотека, для якої обчислюються параметри виходячи із множини авторів. У сховище даних зберігаються серіалізовані об’єкти авторів та бібліотеки, тексти авторів, а також діаграми еталонних грамів та біграмів авторів та бібліотеки. Також, у базу даних зберігаються параметри авторів та бібліотеки. Докладна специфікація структури файлового сховища та бази даних наведена у розділі 2 – Інформаційне забезпечення.

Після створення бібліотеки система готова до основного циклу використання, а саме до аналізу невідомих текстів та визначення параметрів їхніх авторів.

Діаграма послідовності описаних етапів наведена на рисунку 4.2

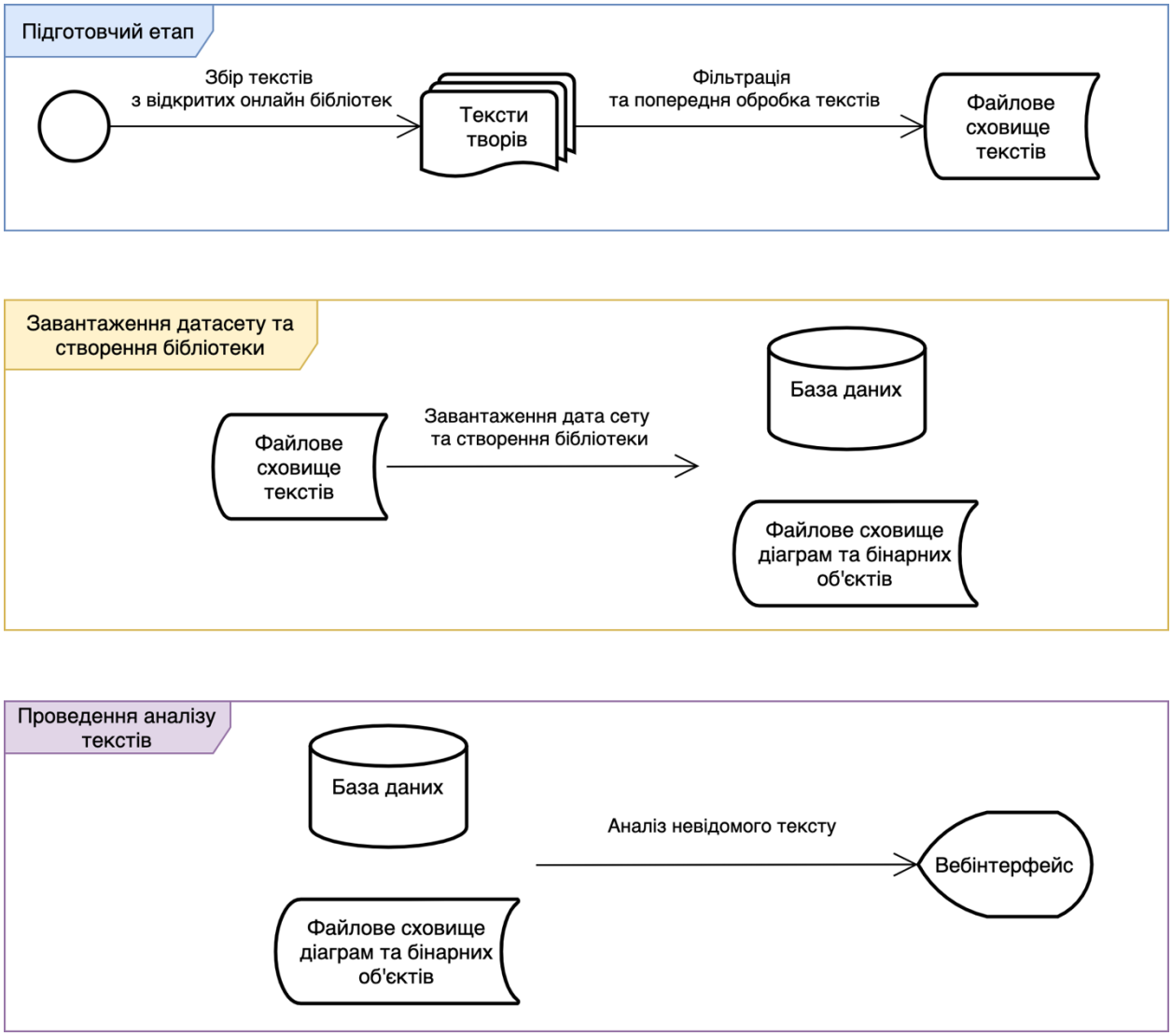


Рисунок 4.2 – Діаграма послідовності етапів використання системи

4.7 Діаграма компонентів

Розроблена система має клієнт-серверну архітектуру. Клієнтська частина реалізована за допомогою вебінтерфейсу, з якою користувач взаємодіє за допомогою веббраузеру.

Серверна частина складається з двох компонентів, вебсерверу та серверу баз даних. Між собою ці компоненти обмінюються даними через віртуальну локальну мережу. Зовнішній доступ можливий лише до вебсерверу за допомогою протоколу HTTP, для користувача та протоколу SSH для адміністратора. Завдяки тому, що доступ до серверу баз даних можливий лише через консоль вебсерверу, а доступ до вебсерверу обмежений лише двома портами, досягається високий рівень безпеки системи.

На рисунку 4.3 наведено діаграму компонентів системи.

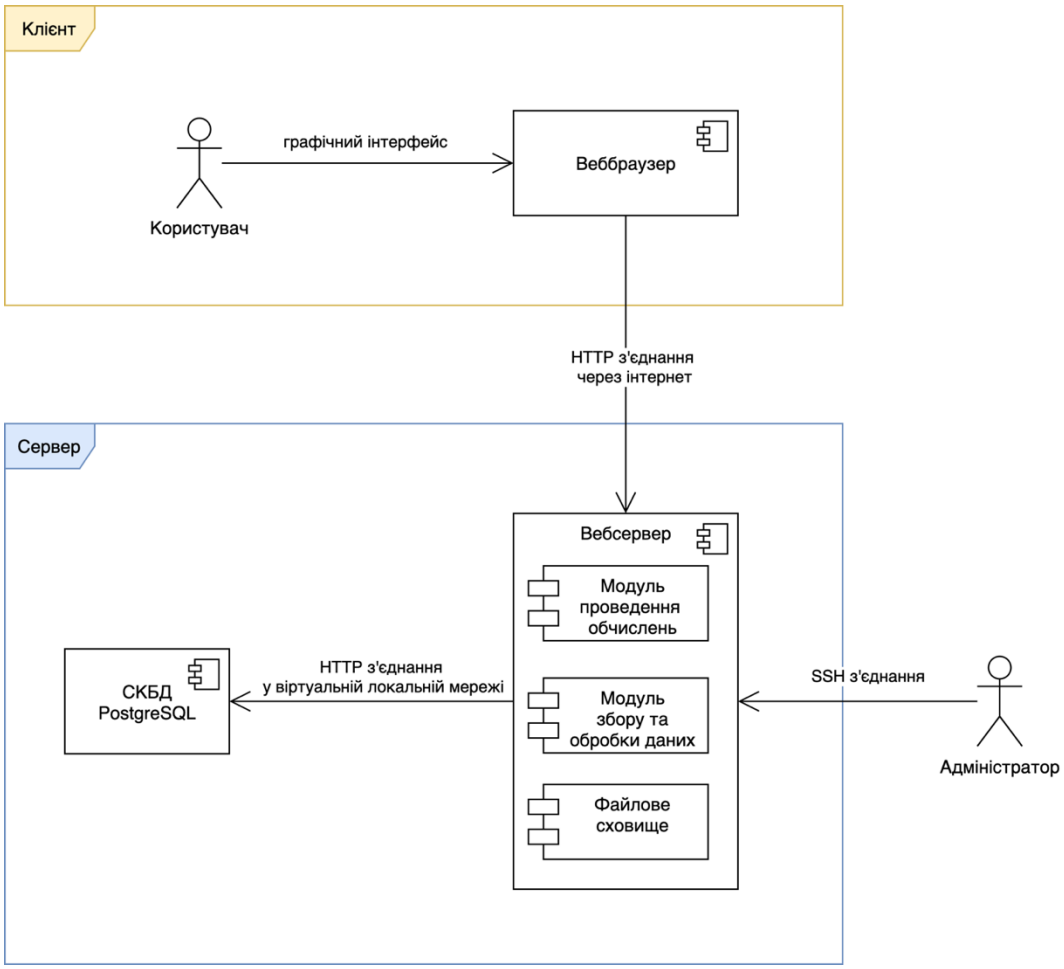


Рисунок 4.3 – Діаграма компонентів системи

4.8 Специфікація функцій

Оскільки система спроектована з використанням парадигми об’єктно-орієнтованого програмування, основна частина функцій, що виконуються в процесі її експлуатаціє є методами відповідних класів.

Отже, наведемо докладний опис усіх публічних методів для кожного з чотирьох класів: Gramm, Text, Author, Library. В таблицях 4.1–4.4 задокументовано назви та призначення кожного з публічних методів відповідного класу.

Таблиця 4.1 – Специфікація публічних функцій класу Gram

Функція	Призначення
add(self, other)	Додає грам other до self
distance_to(self, other)	Повертає список відстаней між грамами other та self
get_all(self)	Повертає словники грамів self усіх порядків
get_n(self, n)	Повертає словник граму self порядку n
remove_all_but_letters(text)	Повертає text у якому містяться лише літери українського алфавіту у нижньому регістрі
draw(self, title, folder)	Зберігає діаграми для граму та біграму self з зазначенням назви title у папці folder

Таблиця 4.2 – Специфікація публічних функцій класу Text

Функція	Призначення
calculate_distance_to_author(self, gram)	Обраховує список відстаней між грамами тексту self та автора gram
remove_punctuation(text)	Повертає text очищеним від знаків пунктуації

Таблиця 4.3 – Специфікація публічних функцій класу Author

Функція	Призначення
add_text(self, text)	Додає текст text до self
read_texts(self)	Завантажує тексти автора з файлового сховища
read_info(self)	Завантажує інформацію про автора з файлового сховища

Продовження таблиці 4.3

Функція	Призначення
calculate_distance_to_library(self, gram)	Обраховує список відстаней між грамами автора self та бібліотеки gram
clean_vocabulary(self, common_words)	Очищає словник автора self від слів, спільних для всіх авторів common_words
save(self)	Зберігає серіалізований об’єкт автора self, його тексти та діаграми, до файлового сховища
load(self)	Завантажує серіалізований об’єкт автора self з файлового сховища

Таблиця 4.4 – Специфікація публічних функцій класу Library

Функція	Призначення
add_author(self, author)	Додає автора author до self
load_authors(self)	Завантажує авторів з файлового сховища
update_library(self, a_names)	Поновлює параметри бібліотеки
update_authors(self)	Поновлює параметри авторів
closest_authors(gram)	Повертає список авторів, найближчих до gram з присутніх у бібліотеці
save(self)	Зберігає серіалізований об’єкт бібліотеки self, його тексти та діаграми, до файлового сховища
load(self)	Завантажує серіалізований об’єкт бібліотеки self з файлового сховища

Висновок до розділу

У ході написання даного розділу дипломного проєкту було розглянуто засоби розробки програмного забезпечення, та обґрунтовано вибір мови програмування Python, а також системи управління базами даних PostgreSQL.

Окрім цього були виявлені та описані загальні вимоги до технічного та програмного забезпечення, які потрібні для використання та коректної роботи системи. Вимоги до технічного забезпечення включають перелік необхідних для запуску застосунку технічних вимог та загальні вимоги до експлуатації.

Було описано обрану для реалізації проекту клієнт-серверну архітектуру системи та обґрунтовано її вибір. Спроектовано діаграми класів, послідовності та компонентів і наведено їх детальний опис. Також наведено специфікацію функцій, що використовувалися при створенні програмного продукту.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Відповідно до розділу 1.1.2 Опис функціональної системи, користувач має змогу виконувати такі функції за допомогою інтерфейсу системи:

- Переглянути параметри бібліотеки;
- Переглянути перелік авторів;
- Переглянути параметри автора;
- Переглянути тексти автора;
- Проаналізувати текст;
- Переглянути перелік звітів;
- Переглянути звіт.

Для забезпечення цієї функціональності було розроблено вебінтерфейс системи, що складається з чотирьох розділів:

- Бібліотека;
- Автори;
- Звіти;
- Аналізувати текст.

Для роботи з системою, користувачу необхідно відкрити веббраузер та перейти за адресою, на якій запущено серверну складову системи. Якщо система запущено на локальному комп’ютері користувача, такою адресою буде http://localhost:5000, у разі ж якщо систему запущено на віддаленому сервері, необхідно ввести адресу відповідного серверу.

Головною сторінкою інтерфейсу системи є розділ Бібліотека. Він надає можливість переглянути параметри бібліотеки та містить таку інформацію:

- Назва завантаженого корпусу текстів;
- Гістограму еталонного граму бібліотеки;
- Теплову карту еталонного біграму бібліотеки;
- Параметри бібліотеки: кількість авторів, текстів, слів та символів, що містяться в бібліотеці;
- Перелік слів, що присутні у кожного з авторів, наявних у бібліотеці.

Рисунок 5.1 – Розділ Бібліотека

Розділ Автори надає можливість переглянути перелік авторів, присутніх у системі та перейти на сторінку Автора. Основна сторінка розділу містить таку інформацію для кожного з авторів:

- Кількість текстів;
- Кількість слів;
- Кількість символів;
- Відстань до еталонних n-грамів бібліотеки.

Автори

Ім'я	Текстів	Слів	Символів	Відстань до n-грамів бібліотеки
Андієвська Емма	2	92 208	480 764	1: 0.053 2: 0.155 3: 0.342 4: 0.623 5: 1.016
Андріяшик Роман	5	318 389	1 651 473	1: 0.038 2: 0.094 3: 0.186 4: 0.356 5: 0.653
Андрухович Юрій	6	153 300	813 906	1: 0.043 2: 0.107 3: 0.234 4: 0.463 5: 0.829
Антоненко-Давидович Борис	47	665 478	3 494 060	1: 0.033 2: 0.090 3: 0.185 4: 0.344 5: 0.600
Арєнєв Володимир	4	155 443	798 323	1: 0.042 2: 0.114 3: 0.254 4: 0.487 5: 0.844
Багмут Іван	5	111 932	566 128	1: 0.058 2: 0.140 3: 0.301 4: 0.560 5: 0.939
Багряний Іван	23	819 873	4 115 568	1: 0.031 2: 0.089 3: 0.193 4: 0.356 5: 0.615
Барка Василь	8	180 497	993 147	1: 0.080 2: 0.184 3: 0.336 4: 0.564 5: 0.906
Бердник Олесь	37	1 322 172	7 079 252	1: 0.044 2: 0.109 3: 0.220 4: 0.380 5: 0.590
Бережний Василь	59	432 986	2 282 674	1: 0.030 2: 0.096 3: 0.222 4: 0.424 5: 0.718
Білецький Олександр	4	49 155	286 640	1: 0.097 2: 0.257 3: 0.491 4: 0.826 5: 1.234
Білий Дмитро	1	44 644	244 044	1: 0.070 2: 0.169 3: 0.352 4: 0.676 5: 1.115
Білик Іван	11	832 920	4 232 287	1: 0.053 2: 0.134 3: 0.267 4: 0.445 5: 0.686
Бічуя Ніна	23	349 706	1 759 389	1: 0.030 2: 0.093 3: 0.210 4: 0.395 5: 0.686

Рисунок 5.2 – Розділ Автори

Сторінка кожного з авторів містить такий перелік інформаційних блоків:

- Прізвище та ім’я автора;
- Роки життя;
- Посилання на сторінки з біографією автора;
- Регіони проживання автора;
- Гістограму еталонного граму автора;
- Теплову карту еталонного біграму автора;
- Перелік текстів, для кожного з яких зазначено: назву, кількість слів, кількість символів та відстань до еталонних n-грамів автора;
- Словник з 50 найуживаніших автором слів.

Рисунок 5.3 – Сторінка автора

Розділ Звіти надає можливість переглянути перелік звітів щодо аналізу невідомих текстів, проведених за допомогою системи. Основна сторінка розділу містить таку інформацію для кожного з авторів:

- Порядковий номер звіту;
- Назва звіту;
- Кількість слів у тексті, що аналізувався;
- Кількість символів у тексті що аналізувався.

Рисунок 5.4 – Розділ Звіти

Сторінка кожного зі звітів містить такий перелік інформаційних блоків:

- Гістограму частотності літер у тексті, що аналізувався;
- Теплову карту біграму тексту, що аналізувався;
- Параметри тексту, що аналізувався: назву, кількість слів та кількість символів;
- Перелік найближчих авторів, для кожного з n-грамів;
- Текст твору, що аналізувався.

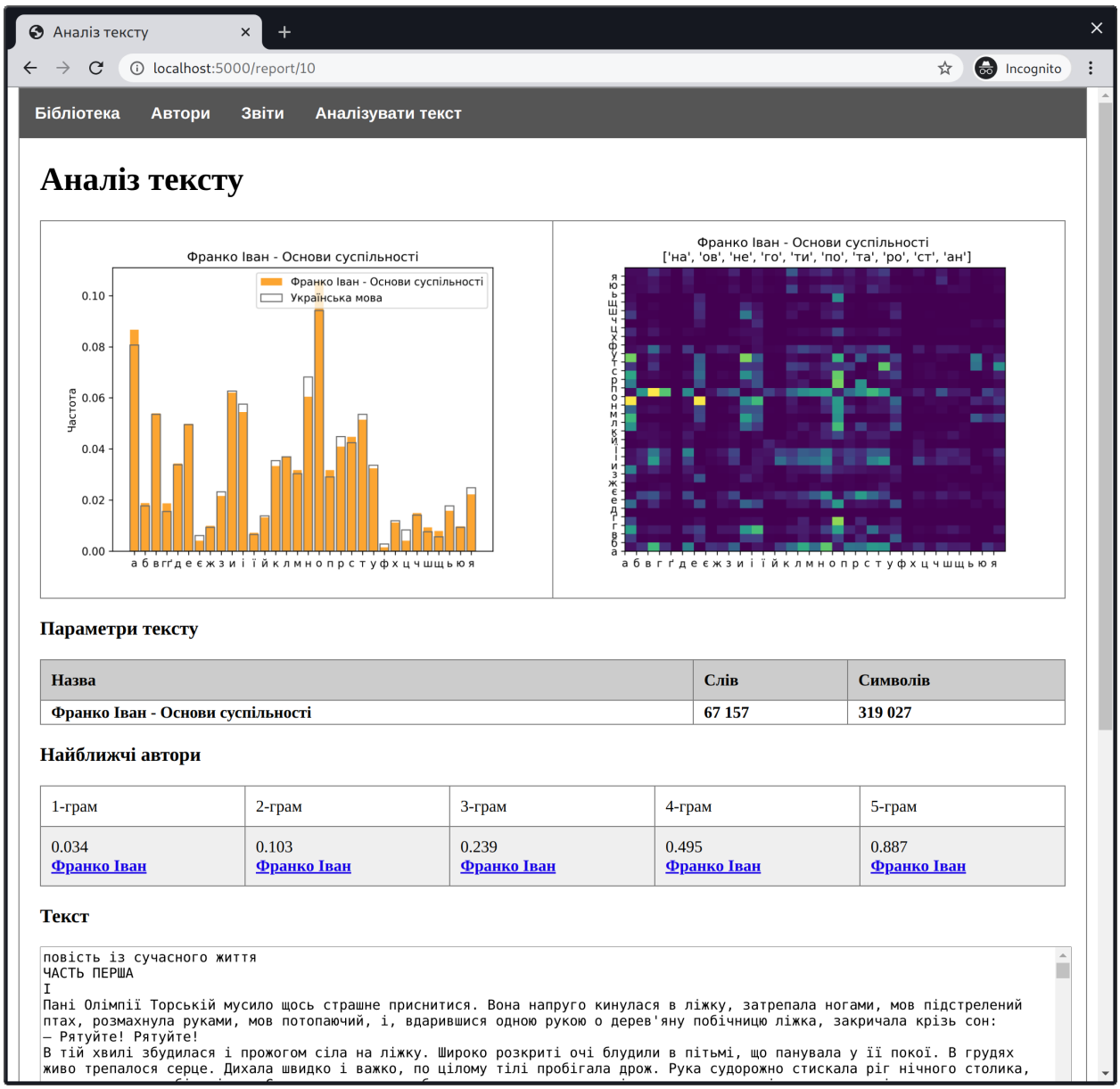


Рисунок 5.5 – Сторінка звіту

Розділ аналізу текстів надає можливість завантажити для аналізу текст невідомого автора. Для проведення тесту необхідно ввести назву тексту, її буде використано як назву відповідного тесту, скопіювати текст у відповідне поле та натиснути кнопку Аналізувати. Після завершення аналізу, швидкість виконання якого суттєво відрізнятиметься в залежності від кількості авторів у бібліотеці та потужності серверу, користувача буде переадресовано на сторінку з звіту з результатами аналізу.

Рисунок 5.6 – Розділ Аналізувати текст

5.2 Випробування програмного продукту

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення комплексу задач функціональним вимогам, представленим у технічному.

В попередньому розділі наведено зображення інтерфейсу системи та перелічено функціональні можливості кожної сторінки інтерфейсу системи. Таким чином, ми бачимо що усі функціональні вимоги, поставлені перед

системою було успішно реалізовано.

Отже, нам лише залишається провести випробування щодо аналітичних властивостей алгоритму визначення параметрів автора.

5.3 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач системи визначення параметрів автора за текстами творів вимогам технічного завдання.

5.4 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.5 Результати випробувань

Для проведення випробувань було підібрано набір текстів авторів, присутніх та відсутніх у системи, з метою отримання усіх можливих конфігурацій результатів проведення аналізу та атрибуції тексту.

Випробування 1.

Автор твору присутній в бібліотеці, текст твору відсутній у бібліотеці.

Як бачимо з результатів звіту, наведених на рисунку 5.7, відбулася коректна атрибуція авторства тексту за кожним з n-грамів.

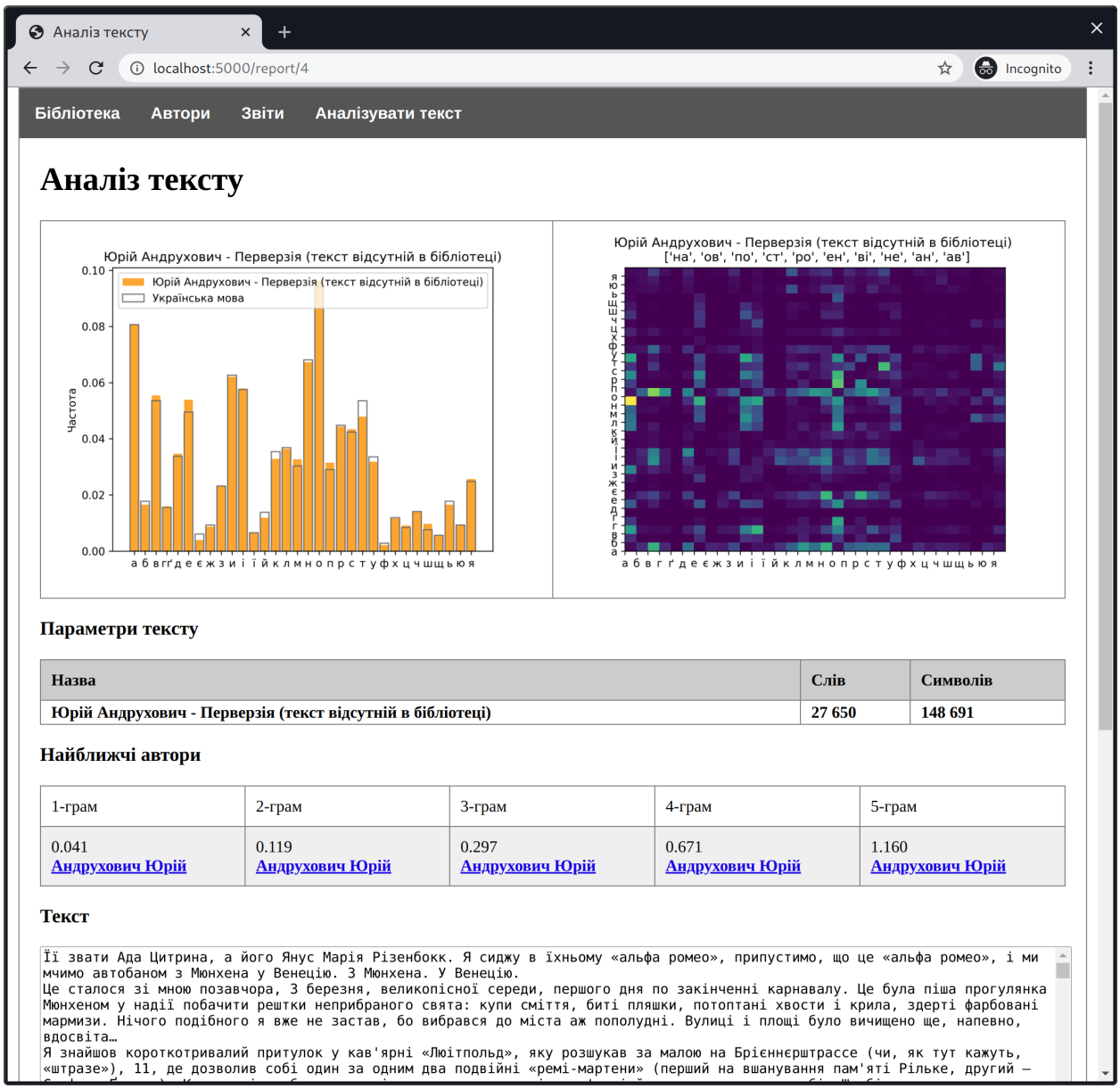


Рисунок 5.7 – Результати випробування 1

Випробування 2.

Автор твору відсутній в бібліотеці.

Як бачимо з результатів звіту, наведених на рисунку 5.8, за кожним з n-грамів було знайдено найближчого автора. У даному випадку обидва автори сучасні, живуть та працюють у Львові, їхні тексти мають багато запозичень з польської та німецької, а отже вони збігаються за стилістичними параметрами.

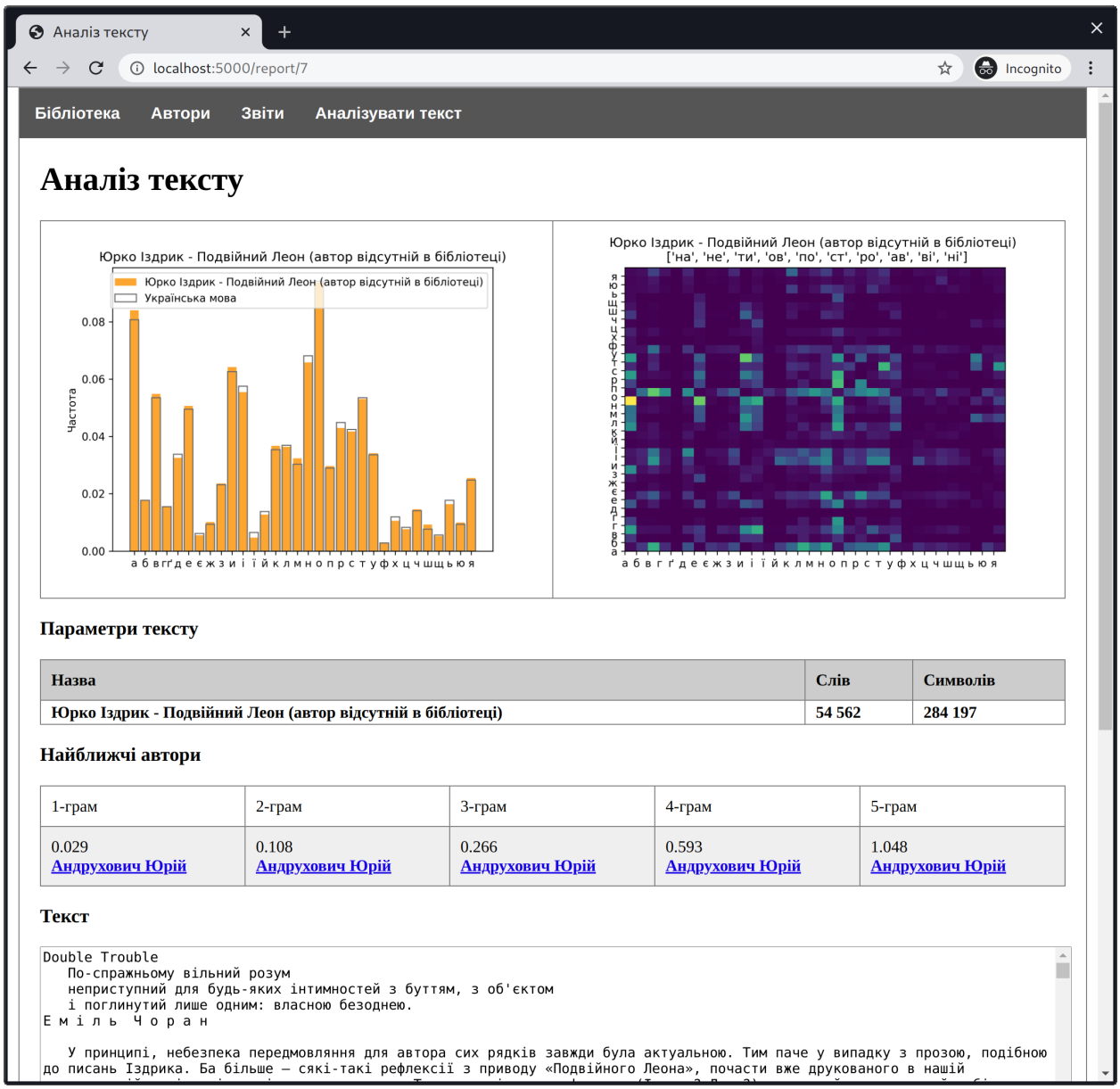


Рисунок 5.8 – Результати випробування 2

Випробування 3.

Автор твору відсутній в бібліотеці.

Ми знову аналізуємо текст сучасної авторки, яка живе і працює у Львові, але не використовує запозичені іншомовні слова у великій кількості. Як бачимо з результатів звіту, наведених на рисунку 5.9, в бібліотеці було знайдено трьох сучасних авторів, які мають схожу стилістику написання творів.

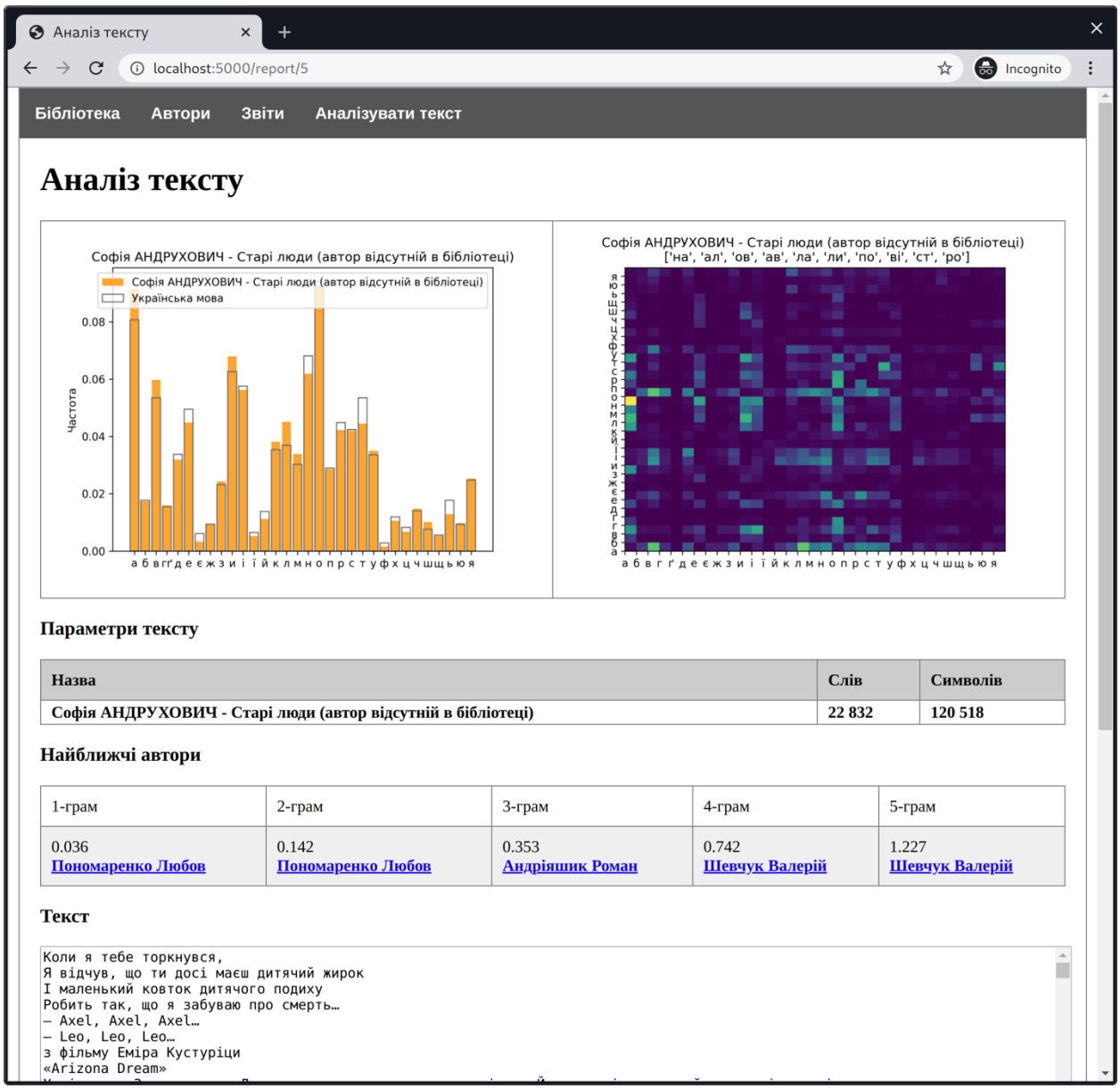


Рисунок 5.9 – Результати випробування 3

Випробування 4.

Для проведення наступного випробування візьмемо текст, який містяться в бібліотеці, але має велику відстань до еталонного граму автора.

Як бачимо з результатів звіту, наведених на рисунку 5.10, відбулася помилкова атрибуція авторства тексту за грамами порядків від 1 до 3, в той же час грами більш високих порядків дали змогу визначити справжнього автора. Таким чином, ми бачимо що грами більш високих порядків не тільки містять інформацію про стилістичні особливості текстів автора, але й допомагають

коректніше визначити авторство.

Також в цьому випробуванні видно, що усі найдені автори належать до однієї епохи.

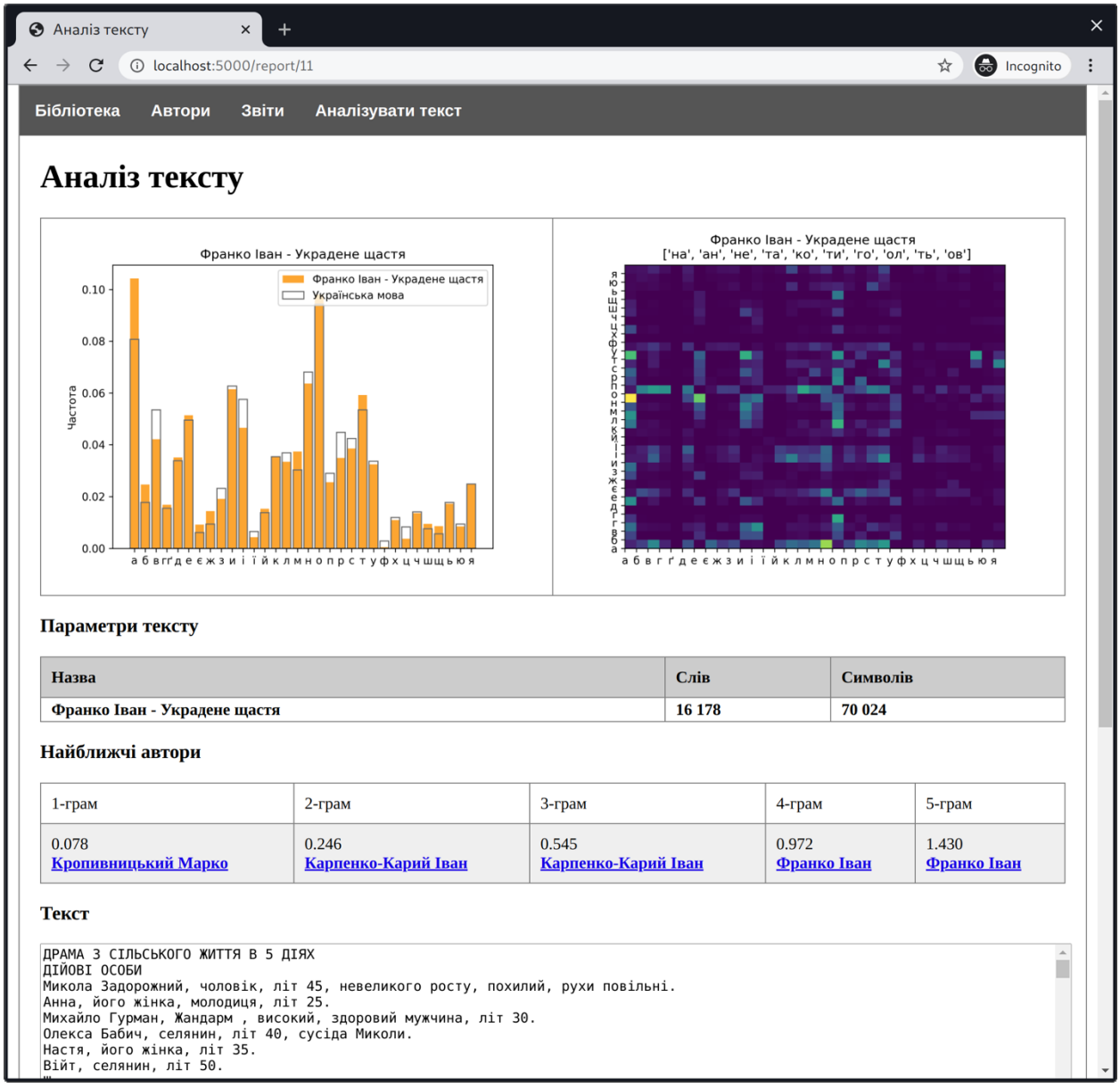


Рисунок 5.10 – Результати випробування 4

Висновок до розділу

Під час написання розділу дипломного проєкту, було сформовано керівництво користувача та наведено екранні форми клієнтського інтерфейсу розробленої системи.

Було встановлено мету проведення випробувань, наведено загальні

положення. Також, було проведено набір репрезентативних випробувань.

У процесі тестування було перевірено функціональність системи, щодо можливостей атрибуції авторства текстів та було встановлено, що функціонал розробленої системи відповідає встановленим вимогам та реалізовує усі функціональні вимоги, висунуті до системи у першому розділі.

ЗАГАЛЬНІ ВИСНОВКИ

Оскільки більшість інформації в наші дні представлена у текстовому форматі, існує необхідність у створенні автоматизованих систем обробки текстових даних. Задача визначення параметрів автора за текстами творів є актуальною у багатьох сферах, але досі не є формалізованою для отримання гарантованого результату щодо визначення авторства, тому існує необхідність у продовженні досліджень для вирішення поставленої задачі.

Під час виконання дипломного проєкту було досліджено предметну область, визначено вимоги до застосування та доведена актуальність розробки. Визначені користувачі системи та варіанти використання застосунку. Виявлено необхідні вхідні дані та їх джерела. Наведено структуру та спосіб відображення вихідних даних. За допомогою діаграм варіантів використання та креслень екранних форм докладно описано та зображено реалізовані у застосунку функції та процеси.

Під час аналізу предметної області була сформульована задача розробки системи. Визначені основні цілі розробки системи визначення параметрів автора за текстами творів. Також визначено задачі, що необхідно було виконати для успішного функціонування системи.

Було розглянуто засоби розробки програмного забезпечення, обґрунтовано вибір мови програмування Python та мікровебфреймворку Flask, а також системи керування базами даних PostgreSQL.

Окрім цього були виявлені загальні вимоги до програмного забезпечення, описано бібліотеки та фреймворк, що використовувались для розробки програмного продукту.

Було описано архітектуру програмного забезпечення. Спроектовано діаграми класів, послідовності та компонентів і наведено їх детальний опис.

Також наведено специфікацію функцій, що використовувалися при створенні програмного продукту та описано звіти, що генеруються в результаті роботи програми.

Також було зібрано корпус текстів літературних творів українських

авторів, на основі яких було побудовано еталонні n-грами для вирішення задачі атрибуції авторства.

Отже, цілі, що були поставлені на першому етапі проектування системи визначення параметрів автора за текстами творів були досягненими в ході виконання дипломного проєкту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Богдан О. Текст і автор: до питання про співвідношення понять / Богдан О., Дмитренко, Є. – Вісник Харківського національного університету імені В. Н. Каразіна. Серія «Філологія», 71(1127), 296-301. – 2015. – Режим доступу до ресурсу: https://periodicals.karazin.ua/philology/article/view/2027
2. Батура Т.В. Формальные методы установления авторства текстов и их реализация в программных продуктах / Батура Т.В. – Институт систем информатики им. А.П. Ершова СО РАН / журнал Программные продукты и системы № 4 – 2013. – Режим доступу до ресурсу: http://www.swsys.ru/index.php?id=3703&page=article
3. Романов А.С. Методика и программный комплекс для идентификации автора неизвестного текста: Автореф. дис. канд. техн. наук. / Романов А.С. – Томск. – 2010. – 26 с.
4. Борисов Л.А. Идентификация автора текста по распределению частот буквосочетаний / Борисов Л.А., Орлов Ю.Н., Осминин К.П. – Препринты ИПМ им. М.В. Келдыша. – 2013. № 27. – 26 с. – Режим доступу до ресурсу: http://library.keldysh.ru/preprint.asp?id=2013-27
5. Орлов Ю.Н. Методы статистического анализа литературных текстов / Орлов Ю.Н., Осминин К.П. – М.: Эдиториал УРСС/Книжный дом «ЛИБРОКОМ» – 2017. – 312 с.
6. Бібліотека української літератури [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://ukrlib.com.ua
7. Хмелёв Д.В. Распознавание автора текста с использованием цепей А.А. Маркова. / Хмелёв Д.В. – Вестник МГУ, сер.9: Филология, N02. – 2000. – с.115–126.
8. Тимашев А.Н. Программа Атрибутор [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://www.textology.ru/atr_resum.html
9. Рогов А.А., Гурин Г.Б., Котов А.А., Сидоров Ю.В., Суровцова Т.Г. Программный комплекс СМАЛТ // Электронные библиотеки:

перспективные методы и технологии, электронные коллекции: Труды X Всерос. науч. конф. «RCDL'2008». Дубна. – 2008. – С. 155–160.

10. Архипова О.О. Частотний аналіз використання букв української мови / Архипова О.О., Журавльов В.М. – Радіоелектроніка, інформатика, управління. – Запоріжжя, 2009. № 2. – С. 53–56 – Режим доступу до ресурсу: https://cyberleninka.ru/article/v/chastotniy-analiz-vikoristannya-bukv-ukrayinskoyi-movi
11. Baudouin C. Elements de cryptographie / C. Baudouin, Ed. A. Pedone. – Paris, 1939. – 214 p.
12. Перебийніс В.С. Статистичні методи для лінгвістів: Навчальний посібник / Перебийніс В.С. — Вінниця: Нова книга, 2002. — 168 с.
13. Сушко С.О. Частоти повторюваності букв і біграм у відкритих текстах українською мовою / Сушко С.О., Фомичова Л.Я., Барсуков Є.С. – 2010. – Режим доступу до ресурсу: http://jrnl.nau.edu.ua/index.php/ZI/article/viewFile/1968/1959
14. Архипова О.О. Артикуляційні таблиці слів української мови / Олена Архипова О.О., Журавльов В.М., Кумейко В.В. – Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, 2(19) вип. – 2009. – С. 13–17. – Режим доступу до ресурсу: http://aprodeus.narod.ru/Teaching/ZAI/Art_2009_Arhipova_Artikul_tabl_ukr_lang.pdf
15. The Python Tutorial [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.python.org/3/tutorial/index.html
16. Serious Python: Black-Belt Advice on Deployment, Scalability, Testing, and More / Julien Danjou – No Starch Press, Inc – 2018. – 240 pp.
17. Flask's documentation [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://flask.palletsprojects.com
18. PostgreSQL documentation [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.postgresql.org/about/
19. SQLAlchemy documentation [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.sqlalchemy.org/

Додаток А

Тексти програмного коду

Система визначення параметрів автора за текстами творів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

38 арк, 256 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

./.env

```
# Database settings
DB_HOST=localhost
DB_NAME=library
DB_USER_NAME=text_analyzer
DB_USER_PASSWORD=PaSSw0rd

# Flask settings
FLASK_ENV=development

SECRET_KEY=6eac@MWW9nwsZa*Qi9*

BLOBS=/blobs/
PUBLIC=/app/static/authors/
UPLOADS=/app/static/uploads/
REPORTS=/app/static/reports/
```

./config.py

```
import multiprocessing
import os
from dotenv import load_dotenv
from logging import DEBUG, INFO, WARNING, ERROR, CRITICAL

LOG_LEVEL = INFO

# Max level of n-gram to generate
N_GRAM = 5

ALPHABET = 'абвгґдеєжзиіїйклмнопрстуфхцчшщьюя'

# Average frequency for letters in Ukrainian language
ALPHA_FREQ = {'а': 0.0807, 'б': 0.0177, 'в': 0.0535, 'г': 0.0155, 'д': 0.0338, 'е': 0.0495, 'є':
0.0061, 'ж': 0.0093,
        'з': 0.0232, 'и': 0.0626, 'і': 0.0575, 'ї': 0.0065, 'й': 0.0138, 'к': 0.0354, 'л': 0.0369, 'м':
0.0303,
```

```
        'н': 0.0681, 'о': 0.0942, 'п': 0.0290, 'р': 0.0448, 'с': 0.0424, 'т': 0.0535, 'у': 0.0336, 'ф': 0.0028,
        'х': 0.0119, 'ц': 0.0083, 'ч': 0.0141, 'ш': 0.0076, 'щ': 0.0056, 'ь': 0.0177, 'ю': 0.0093, 'я': 0.0248}

load_dotenv()

N_PROC = multiprocessing.cpu_count()

# Storage directories
ROOT_DIR = os.path.dirname(os.path.abspath(__file__))
BLOBS = f'{ROOT_DIR}{os.getenv("BLOBS")}'
PUBLIC = f'{ROOT_DIR}{os.getenv("PUBLIC")}'
UPLOADS = f'{ROOT_DIR}{os.getenv("UPLOADS")}'
REPORTS = f'{ROOT_DIR}{os.getenv("REPORTS")}'



# Flask app configuration
class Config:
    """Base configuration settings for Flask app"""
    SECRET_KEY = os.getenv('SECRET_KEY')

    DB_HOST = os.getenv('DB_HOST')
    DB_NAME = os.getenv('DB_NAME')
    DB_USER_NAME = os.getenv('DB_USER_NAME')
    DB_USER_PASSWORD = os.getenv('DB_USER_PASSWORD')

    SQLALCHEMY_DATABASE_URI = f'postgresql+psycopg2://{DB_USER_NAME}:{DB_USER_PASSWORD}@{DB_HOST}/{DB_NAME}'
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

./import_uploads.py

```
import logging

import config as cfg
```

```
from models import Library

if __name__ == '__main__':
    logging.basicConfig(
        format='%(asctime)s %(levelname)s:%(module)s:%(message)s',
        level=cfg.LOG_LEVEL,
        datefmt='%H:%M:%S'
    )

    logging.info('start')
    library = Library()
    library.load_authors()
    library.save()
    logging.info('finish')
```

./models/__init__.py

```
from .author import Author
from .gram import Gram
from .library import Library
from .text import Text
```

./models/gram.py

```
import os
import re
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter
from multiprocessing import Process

import config as cfg


class Gram:
    def __init__(self, text=''):
        self.n_letters = 0
```

```
        self._gram = [Counter() for _ in range(cfg.N_GRAM)]
        self._generate_grams(text, n=cfg.N_GRAM)

    def add(self, other):
        """Add given gram to self."""
        for n in range(cfg.N_GRAM):
            self._gram[n] += other._gram[n]
        self.n_letters += other.n_letters

    def distance_to(self, other):
        """Calculate distances to given gram."""
        self_frq = self.get_all()
        other_frq = other.get_all()
        return [Gram._distance(self_frq[n], other_frq[n]) for n in range(cfg.N_GRAM)]

    def get_all(self):
        """Return all grams with frequencies."""
        return [self.get_n(n) for n in range(cfg.N_GRAM)]

    def get_n(self, n):
        """Return n-gram with frequencies."""
        return {i[0]: (i[1] / self.n_letters) for i in self._gram[n].items()} if self.n_letters else None

    def draw(self, title, folder):
        """Save gram and bigram images to given folder."""
        Gram._draw_grams(self.get_n(0), self.get_n(1), title, folder)

    def _generate_gram(self, letters, n):
        """Generate n-gram with number of occurrences."""
        return Counter([letters[i:i + n] for i in range(self.n_letters - n + 1)])

    def _generate_grams(self, text, n):
        """Generate n-grams."""
        letters = Gram._remove_all_but_letters(text)
        if letters:
            self.n_letters = len(letters)
```

```
        self._gram = [self._generate_gram(letters, i + 1) for i in range(n)]

    def __repr__(self):
        return f'<Gram: [{self.n_letters}]>'

    @staticmethod
    def _remove_all_but_letters(text):
        """Return text in lower case. Leave only letters from ukrainian alphabet."""
        pattern = re.compile(f'[^{cfg.ALPHABET}]')
        return re.sub(pattern, '', text.lower())

    @staticmethod
    def _distance(d1, d2):
        """Return distance between two given dicts."""
        d = 0
        keys = set(d1).union(d2)
        for key in keys:
            d += abs(d1.get(key, 0) - d2.get(key, 0))
        return d

    @staticmethod
    def _draw_grams(gram, bigram, title, folder):
        if not os.path.exists(folder):
            os.makedirs(folder)

        p = Process(target=Gram._draw_gram, args=(gram, title, folder))
        p.start()
        p = Process(target=Gram._draw_bigram, args=(bigram, title, folder))
        p.start()

    @staticmethod
    def _draw_gram(gram, title, folder):
        gram['т'] += gram.get('ґ', 0)
        x = range(len(cfg.ALPHA_FREQ))
        p1 = plt.bar(x, [gram.get(i, 0) for i in cfg.ALPHA_FREQ], color='orange')
        p2 = plt.bar(x, cfg.ALPHA_FREQ.values(), fill=False, ec='dimgrey')
```

```
        plt.xticks(x, [i if i != 'r' else 'rr' for i in cfg.ALPHA_FREQ])
        plt.title(f'{title}')
        plt.ylabel('Частота')
        plt.legend((p1[0], p2[0]), (f'{title}', 'Українська мова'))
        plt.savefig(f'{folder}gram.png', dpi=300)
        plt.clf()

    @staticmethod
    def _draw_bigram(bigram, title, folder):
        n = len(cfg.ALPHABET)
        m = np.zeros((n, n))

        for i, li in enumerate(cfg.ALPHABET):
            for j, lj in enumerate(cfg.ALPHABET):
                m[i, j] = bigram.get(li + lj, 0)

        top_freq = sorted(bigram.items(), key=lambda x: x[1])[:-11:-1]
        plt.pcolor(m)
        plt.title(f'{title} \n {[i[0] for i in top_freq]}')
        plt.xticks(range(n), cfg.ALPHABET)
        plt.yticks(range(n), cfg.ALPHABET)
        plt.savefig(f'{folder}bigram.png', dpi=300)
        plt.clf()
```

./models/text.py

```
import re
from collections import Counter

import config as cfg
from .gram import Gram


class Text:
    def __init__(self, title, text):
        self.title = title
        self.text = text
```

```
        self.n_words = 0
        self.gram = Gram(text)
        self.vocabulary = Counter()
        self.distance_to_author = []
        self._generate_vocabulary(self.text)

    @property
    def n_letters(self):
        return self.gram.n_letters

    def calculate_distance_to_author(self, gram):
        """Calculate distances from text to given author gram."""
        self.distance_to_author = self.gram.distance_to(gram)

    def _generate_vocabulary(self, text):
        """Generate vocabulary with number of occurrences."""
        words = Text.remove_punctuation(text).split()
        if words:
            self.n_words = len(words)
            self.vocabulary = Counter(words)

    def __repr__(self):
        return f'<Text: {self.title} [{self.n_letters}]>'

    @staticmethod
    def remove_punctuation(text):
        pattern = re.compile(f'[^{cfg.ALPHABET} \n]')
        return re.sub(pattern, '', text.lower())
```

./models/author.py

```
import logging
import os
import pickle
from datetime import datetime
from collections import Counter
from pathlib import Path
```

```
import yaml

import config as cfg
from app import db
from app.tables import AuthorTable, TextTable
from .gram import Gram
from .text import Text


class Author:
    def __init__(self, first_name, last_name):
        logging.info(f'Creating Author instance: {first_name} {last_name}')
        self.first_name = first_name
        self.last_name = last_name
        self.date_birth = None
        self.date_death = None
        self.location = None
        self.url_ukrlib = None
        self.url_wiki = None
        self.n_words = 0
        self.gram = Gram()
        self.vocabulary = Counter()
        self.texts = []
        self.distance_to_library = []

    @property
    def name(self):
        return f'{self.last_name} {self.first_name}'

    @property
    def n_texts(self):
        return len(self.texts)

    @property
    def n_letters(self):
```

```
        return self.gram.n_letters

    def add_text(self, text):
        """Add text. Update gram and vocabulary."""
        logging.debug(f'.add_text: {text} to {self}')
        if isinstance(text, Text):
            self.texts.append(text)
            self.n_words += text.n_words
            self.vocabulary += text.vocabulary
            self.gram.add(text.gram)
        else:
            raise TypeError(f'Expected {type(self)}, got {type(text)} instead.')

    def read_texts(self):
        """Read texts from given folder and add to author's texts."""
        logging.debug(f'.read_texts: {self}')
        folder = f'{cfg.UPLOADS}{self.name}/'
        for file in [i for i in Path(folder).iterdir() if i.is_file() and i.name.endswith('.txt')]:
            with open(str(file)) as f:
                text_body = f.read()
            text_title = file.name[:-4]
            text = Text(title=text_title, text=text_body)
            self.add_text(text)

    def read_info(self):
        folder = f'{cfg.UPLOADS}{self.name}/'
        info_yml = f'{folder}info.yml'
        if os.path.exists(info_yml):
            with open(info_yml) as info:
                author_info = yaml.load(info, Loader=yaml.Loader)
            self.date_birth = author_info.get('date_birth')
            self.date_death = author_info.get('date_death')
            self.location = author_info.get('location')
            self.url_ukrlib = author_info.get('url_ukrlib')
            self.url_wiki = author_info.get('url_wiki')
```

```
    def calculate_distance_to_library(self, gram):
        """Calculate distances from author to given library gram."""
        self.distance_to_library = self.gram.distance_to(gram)

    def clean_vocabulary(self, common_words):
        """Remove common_words from author's vocabulary."""
        self.vocabulary = {key: value for key, value in sorted(self.vocabulary.items(), key=lambda
x: x[1])
                           if key not in common_words}

    def save(self):
        """Save instance components to disk and db."""
        logging.debug(f'.save: {self}')
        self._calculate_distances_from_texts()
        self._save_self()
        self._save_images()
        self._save_texts()
        self._save_to_db()

    def load(self):
        """Load author from blob."""
        logging.debug(f'.load: start {self}')
        with open(f'{cfg.BLOBS}{self.name}/author.pickle', 'br') as f:
            d = pickle.load(f)
        self.__dict__.clear()
        self.__dict__.update(d)
        logging.debug(f'.load: finish {self}')

    def _save_self(self):
        """Save serialized self instance to blobs folder."""
        logging.debug(f'._save_self: {self}')
        folder = f'{cfg.BLOBS}{self.name}/'
        if not os.path.exists(folder):
            os.makedirs(folder)

        with open(f'{folder}author.pickle', 'bw') as f:
```

```
            pickle.dump(self.__dict__, f)

    def _save_images(self):
        """Save gram and bigram images to public folder."""
        logging.debug(f'._save_images: {self}')
        folder = f'{cfg.PUBLIC}{self.name}/images/'
        self.gram.draw(self.name, folder)

    def _save_texts(self):
        """Save texts to public folder."""
        logging.debug(f'._save_texts: {self}')
        folder = f'{cfg.PUBLIC}{self.name}/'
        for text in self.texts:
            with open(f'{folder}{text.title}.txt', 'w') as f:
                f.write(text.text)

    def _save_to_db(self):
        """Save author to db."""
        logging.debug(f'._save_db: {self}')
        a = AuthorTable(
            first_name=self.first_name,
            last_name=self.last_name,
            date_birth=datetime.strptime(str(self.date_birth), '%Y') if self.date_birth else None,
            date_death=datetime.strptime(str(self.date_death), '%Y') if self.date_death else None,
            location = self.location,
            url_ukrlib = self.url_ukrlib,
            url_wiki = self.url_wiki,
            n_texts=self.n_texts,
            n_words=self.n_words,
            n_letters=self.n_letters,
            vocabulary=list(sorted(self.vocabulary.items(), key=lambda x: x[1], reverse=True))[:50],
            distance_to_library=self.distance_to_library
        )
        db.session.add(a)
        db.session.commit()
```

```
        for text in self.texts:
            t = TextTable(
                title=text.title,
                text=text.text,
                n_words=text.n_words,
                n_letters=text.n_letters,
                distance_to_author=text.distance_to_author,
                author_id=a.id
            )
            db.session.add(t)
        db.session.commit()

    def _calculate_distances_from_texts(self):
        """Calculate distances from texts to author's average. Update text instances."""
        logging.debug(f'._calculate_distances_from_texts: {self}')
        for text in self.texts:
            text.calculate_distance_to_author(self.gram)

    def __repr__(self):
        return f'<Author: {self.first_name} {self.last_name} [{len(self.texts)}]>'
```

./models/library.py

```
import logging
import os
import pickle
from multiprocessing import Process, Queue
from pathlib import Path

import config as cfg
from app import db
from app.tables import LibraryTable
from .author import Author
from .gram import Gram


class Library:
```

```
def __init__(self, title='Бібліотека'):
    logging.info(f'Creating Library instance: {title}')
    self.title = title
    self.n_texts = 0
    self.n_words = 0
    self.a_names = []
    self.common_words = set()
    self.gram = Gram()

@property
def n_authors(self):
    return len(self.a_names)

@property
def n_letters(self):
    return self.gram.n_letters

def add_author(self, author):
    """Add author. Update grams and vocabulary."""
    logging.info(f'Adding {author} to {self}')
    if isinstance(author, Author):
        self.a_names.append(author.name)
        self.n_texts += author.n_texts
        self.n_words += author.n_words
        self.gram.add(author.gram)
        if not self.common_words:
            self.common_words = set(author.vocabulary)
        self.common_words.intersection_update(author.vocabulary)
    else:
        raise TypeError(f'Expected {type(self)}, got {type(author)} instead.')

def load_authors(self):
    """Read authors from given folder."""
    logging.info(f'.load_authors: {self}')
    a_names = self.read_authors()
    self.update_library(a_names)
```

```
        self.update_authors()

    def read_authors(self):
        logging.info(f'.read_authors: {self}')
        a_names = [i.name for i in Path(f'{cfg.UPLOADS}').iterdir() if i.is_dir() and i.name !=
'library']

        processes = []
        for i in range(cfg.N_PROC):
            p = Process(target=Library._read_authors, args=(a_names[i::cfg.N_PROC],))
            processes.append(p)
            p.start()

        for p in processes:
            p.join()

        return a_names

    @staticmethod
    def _read_authors(a_names):
        logging.info(f'._read_authors: {os.getpid()=}')
        for a_name in a_names:
            last_name, first_name = a_name.split()
            author = Author(last_name=last_name, first_name=first_name)
            author.read_texts()
            author.read_info()
            author._save_self()

    def update_library(self, a_names):
        logging.info(f'.update_authors: {self}')
        for a_name in a_names:
            last_name, first_name = a_name.split()
            author = Author(last_name=last_name, first_name=first_name)
            author.load()
            self.add_author(author)
            author._save_self()
```

```
def update_authors(self):
    logging.info(f'.update_authors: {self}')
    processes = []
    for i in range(cfg.N_PROC):
        p = Process(target=Library._update_authors,
                    args=(self.a_names[i::cfg.N_PROC], self.common_words, self.gram))
        processes.append(p)
        p.start()

    for p in processes:
        p.join()

@staticmethod
def _update_authors(a_names, common_words, gram):
    logging.info(f'._update_authors: {os.getpid()=}')
    for a_name in a_names:
        last_name, first_name = a_name.split()
        author = Author(last_name=last_name, first_name=first_name)
        author.load()
        author.clean_vocabulary(common_words)
        author.calculate_distance_to_library(gram)
        author.save()

def save(self):
    """Save instance components to disk and db."""
    logging.debug(f'.save: {self}')
    self._save_self()
    self._save_to_db()
    self._save_images()

def load(self):
    """Load library from blob."""
    logging.debug(f'.load: start {self}')
    folder = f'{cfg.BLOBS}library/'
    with open(f'{folder}library.pickle', 'br') as f:
```

```
            d = pickle.load(f)
        self.__dict__.clear()
        self.__dict__.update(d)
        logging.debug(f'.load: finish {self}')

    def _save_self(self):
        """Save serialized self instance to blobs folder."""
        logging.debug(f'._save_self: {self}')
        folder = f'{cfg.BLOBS}library/'
        if not os.path.exists(folder):
            os.makedirs(folder)

        with open(f'{folder}library.pickle', 'bw') as f:
            pickle.dump(self.__dict__, f)

    def _save_images(self):
        """Save gram and bigram images to public folder."""
        logging.debug(f'._save_images: {self}')
        folder = f'{cfg.PUBLIC}library/images/'
        self.gram.draw(self.title, folder)

    def _save_to_db(self):
        """Save library to db."""
        logging.debug(f'._save_db: {self}')
        lib = LibraryTable(
            title=self.title,
            n_authors=self.n_authors,
            n_texts=self.n_texts,
            n_words=self.n_words,
            n_letters=self.n_letters,
            common_words=sorted(self.common_words)
        )
        db.session.add(lib)
        db.session.commit()

    def __repr__(self):
```

```
        return f'<Library: {self.title} [{self.n_authors}]>'

    @staticmethod
    def closest_authors(gram, vocabulary):
        folders = [i for i in sorted(Path(f'{cfg.BLOBS}').iterdir(), reverse=True)
                   if i.is_dir() and i.name != 'library']
        processes = []
        q = Queue()
        for i in range(cfg.N_PROC):
            p = Process(target=Library._min_distance, args=(gram, vocabulary,
folders[i::cfg.N_PROC], q))
            processes.append(p)
            p.start()
        for p in processes:
            p.join()

        min_d = q.get()
        while not q.empty():
            tmp_d = q.get()
            for i in range(cfg.N_GRAM):
                if tmp_d[i][1] < min_d[i][1]:
                    min_d[i] = tmp_d[i]
        return min_d

    @staticmethod
    def _min_distance(gram, vocabulary, folders, q):
        min_distance = [2 for _ in range(cfg.N_GRAM)]
        closest_author = ['' for _ in range(cfg.N_GRAM)]

        for folder in folders:
            last_name, first_name = folder.name.split()
            author = Author(last_name=last_name, first_name=first_name)
            author.load()

            for i in range(cfg.N_GRAM):
                if (new_dist := Gram._distance(gram[i], author.gram.get_n(i))) < min_distance[i]:
```

```
                min_distance[i] = new_dist
                closest_author[i] = f'{last_name} {first_name}'
        q.put(list(zip(closest_author, min_distance)))
```

./app/__init__.py

```
from flask import Flask
from flask_migrate import Migrate
from flask_sqlalchemy import SQLAlchemy

from config import Config

app = Flask(__name__)
app.config.from_object(Config)
db = SQLAlchemy(app)
migrate = Migrate(app, db)

from app import routes, tables
```

./app/forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, SubmitField
from wtforms.validators import DataRequired, Length


class TextForm(FlaskForm):
    title = StringField(label='Назва', validators=[DataRequired(message='Це поле обов\'язкове'),
Length(max=255)])
    text = TextAreaField(label='Текст', validators=[DataRequired(message='Це поле
обов\'язкове')])
    submit = SubmitField(label='Аналізувати')
```

./app/routes.py

```
from flask import render_template, flash, redirect
from datetime import datetime
```

```
import config as cfg
from app import app, db
from app.forms import TextForm
from app.tables import AuthorTable, LibraryTable, ReportTable


@app.route('/')
def index_page():
    # TODO add annotation and links to the libraries
    return redirect(f'/library/')


@app.route('/library/')
def library_page():
    library = LibraryTable.query.get(1)
    return render_template('library.html', library=library)


@app.route('/authors/')
def authors_page():
    authors = AuthorTable.query.order_by(AuthorTable.last_name, AuthorTable.first_name)
    return render_template('authors.html', authors=authors)


@app.route('/author/')
@app.route('/author/<name>')
def author_page(name=None):
    if name and len(name.split()) == 2:
        last_name, first_name = name.split()
        author = AuthorTable.query.filter(
            AuthorTable.last_name == last_name,
            AuthorTable.first_name == first_name).first()
    else:
        author = None
    return render_template('author.html', title=name, author=author)
```

```
@app.route('/analyze', methods=['GET', 'POST'])
def analyze_page():
    form = TextForm()
    if form.validate_on_submit():
        from models import Library
        from models import Text
        text = Text(title=form.title.data, text=form.text.data)

        report = ReportTable(
            title=text.title,
            text=text.text,
            n_letters=text.n_letters,
            n_words=text.n_words,
            distance_to_authors=[]
        )
        db.session.add(report)
        db.session.commit()
        text_id = report.id

        folder = f'{cfg.REPORTS}{text_id}/'
        text.gram.draw(text.title, folder)

        min_distance = Library.closest_authors(text.gram.get_all(), text.vocabulary)

        report.distance_to_authors = min_distance
        db.session.commit()

        flash(f'Текст успішно проаналізовано')
        return redirect(f'/report/{text_id}')
    return render_template('analyze.html', form=form)


@app.route('/report/<report_id>', methods=['GET', 'POST'])
def report_page(report_id=None):
```

```
    report = ReportTable.query.get(report_id)
    return render_template('report.html', report_id=report_id, report=report)


@app.route('/reports/')
def reports_page():
    reports = ReportTable.query.order_by(ReportTable.id)
    return render_template('reports.html', reports=reports)


@app.template_filter('year')
def date_format_year(s):
    # Template filter for date fields.
    return datetime.strftime(s, '%Y')
```

./app/tables.py

```
from app import db
from sqlalchemy import UniqueConstraint


class LibraryTable(db.Model):
    """Model for the library table."""
    __tablename__ = 'library'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(128), nullable=False)
    n_letters = db.Column(db.Integer, default=0, nullable=False)
    n_words = db.Column(db.Integer, default=0, nullable=False)
    n_texts = db.Column(db.Integer, default=0, nullable=False)
    n_authors = db.Column(db.Integer, default=0, nullable=False)
    common_words = db.Column(db.PickleType, nullable=True)

    def __repr__(self):
        return f'<LibraryTable [{self.n_authors}]>'
```

```
class AuthorTable(db.Model):
    """Model for the authors table."""
    __tablename__ = 'authors'
    __table_args__ = (
        UniqueConstraint('first_name', 'last_name'),
    )
    id = db.Column(db.Integer, primary_key=True)
    last_name = db.Column(db.String(64), nullable=False)
    first_name = db.Column(db.String(64), nullable=False)
    date_birth = db.Column(db.Date, nullable=True)
    date_death = db.Column(db.Date, nullable=True)
    location = db.Column(db.String(128), nullable=True)
    url_ukrlib = db.Column(db.String(128), nullable=True)
    url_wiki = db.Column(db.String(512), nullable=True)
    n_texts = db.Column(db.Integer, default=0, nullable=False)
    n_words = db.Column(db.Integer, default=0, nullable=False)
    n_letters = db.Column(db.Integer, default=0, nullable=False)
    vocabulary = db.Column(db.PickleType, nullable=True)
    distance_to_library = db.Column(db.PickleType, nullable=True)
    texts = db.relationship('TextTable', backref='authors', passive_deletes='all')

    def __repr__(self):
        return f'AuthorTable <{self.last_name}, {self.first_name}, {self.id=}>'


class TextTable(db.Model):
    """Model for the texts table."""
    __tablename__ = 'texts'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(256), nullable=False)
    text = db.Column(db.Text(), nullable=False)
    n_letters = db.Column(db.Integer, nullable=False)
    n_words = db.Column(db.Integer, nullable=False)
    distance_to_author = db.Column(db.PickleType, nullable=True)
```

```
    author_id = db.Column(db.Integer, db.ForeignKey('authors.id', ondelete='RESTRICT'),
nullable=False)

    def __repr__(self):
        return f'<TextTable ({self.title=}, {self.author_id=})>'


class ReportTable(db.Model):
    """Model for the library table."""
    __tablename__ = 'reports'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(256), nullable=False)
    text = db.Column(db.Text(), nullable=False)
    n_letters = db.Column(db.Integer, default=0, nullable=False)
    n_words = db.Column(db.Integer, default=0, nullable=False)
    distance_to_authors = db.Column(db.PickleType, nullable=True)

    def __repr__(self):
        return f'<ReportTable [{self.id}]>'
```

. /app/templates/analyze.html

```
{% extends "base.html" %}
{% block title %}Аналізувати текст{% endblock %}
{% block menu_active_analyze %}active{% endblock %}
{% block content %}
    <h1>Аналізувати текст</h1>
    <form action="" method="post" novalidate>
        {{ form.hidden_tag() }}
        {{ form.title.label }}<br>{{ form.title(size=64) }}<br>
        {% for error in form.title.errors %}
            <span class="error">{{ error }}</span>
        {% endfor %}
        <br>
        {{ form.text.label }}<br>{{ form.text(rows="40", style="width: 100%") }}<br>
        {% for error in form.text.errors %}
```

```
            <span class="error">{{ error }}</span>
        {% endfor %}
        <p><em>Аналіз може зайняти декілька хвилин, дочекайтесь завантаження сторінки звіту.</em></p>
        {{ form.submit }}
        <button type="reset" onclick="window.location.href = '{{ url_for('analyze_page') }}';">Очистити</button>
    </form>
{% endblock %}
```

./app/templates/author.html

```
{% extends "base.html" %}
{% block title %}{{ title }}{% endblock %}
{% block content %}
    {% if author %}
        <h1>{{ author.last_name }} {{ author.first_name }}
            {% if author.date_birth %}
                ({{ author.date_birth | year }}{% if author.date_death %}&ndash;{{ author.date_death | year }}{% endif %})
            {% endif %}
        </h1>
        {% if author.url_ukrlib or author.url_wiki %}
            <p><b>Біографія:</b>
                {% if author.url_ukrlib or author.url_wiki %}
                    <a href="{{ author.url_ukrlib }}">ukrlib.com.ua</a>,
                {% endif %}
                {% if author.url_ukrlib or author.url_wiki %}
                    <a href="{{ author.url_wiki }}">wikipedia.org</a></p>
                {% endif %}
        {% endif %}
        {% if author.location %}
            <p><b>Географія:</b> {{ author.location }}</p>
        {% endif %}
        <table>
            <tr>
                <td><a href="/static/authors/{{ title }}/images/gram.png"><img
```

```
                src="/static/authors/{{ title }}/images/gram.png" alt="Еталонний грам автора:
{{ title }}"
                width="100%"></a></td>
            <td><a href="/static/authors/{{ title }}/images/bigram.png"><img
                src="/static/authors/{{ title }}/images/bigram.png" alt="Еталонний біграм
автора: {{ title }}"
                width="100%"></a></td>
        </tr>
    </table>
    <h3>Тексти автора</h3>
    <table>
        <thead>
        <tr>
            <th>Назва</th>
            <th>Слів</th>
            <th>Символів</th>
            <th>Відстань до n-грамів автора</th>
        </tr>
        </thead>
        <tbody>
        {% for text in author.texts %}
            <tr>
                <th><p><a href="/static/authors/{{ title }}/{{ text.title }}.txt">{{ text.title
}}</a></p></th>
                <td><b>{{ '{:,}'.format(text.n_words).replace(',', ' ') }}</b></td>
                <td><b>{{ '{:,}'.format(text.n_letters).replace(',', ' ') }}</b></td>
                <td>
                    {% for distance in text.distance_to_author %}
                        {{ loop.index }}: <b>{{ '%0.3f'|format(distance) }}</b>  
                    {% endfor %}
                </td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
    <h3>Найвживаніші слова</h3>
```

```
<table class="no-border">
  <tr>
    <td>
      <table class="words">
        <thead>
        <tr>
          <th>Слово</th>
          <th>Частота</th>
          <th>Кількість</th>
        </tr>
        </thead>
        {% for word in author.vocabulary[:25] %}
          <tr>
            <td><b>{{ word[0] }}</b></td>
            <td>{{ '%0.5f'|format(word[1] / author.n_words) }}</td>
            <td>{{ word[1] }}</td>
          </tr>
        {% endfor %}
      </table>
    </td>
    <td>
      <table>
        <thead>
        <tr>
          <th>Слово</th>
          <th>Частота</th>
          <th>Кількість</th>
        </tr>
        </thead>
        {% for word in author.vocabulary[25:] %}
          <tr>
            <td><b>{{ word[0] }}</b></td>
            <td>{{ '%0.5f'|format(word[1] / author.n_words) }}</td>
            <td>{{ word[1] }}</td>
          </tr>
        {% endfor %}
```

```
            </table>
          </td>
        </tr>
      </table>
    {% else %}
      <h2>Автора {{ title }} не знайдено</h2>
    {% endif %}
{% endblock %}
```

. /app/templates/authors.html

```
{% extends "base.html" %}
{% block title %}Автори{% endblock %}
{% block menu_active_authors %}active{% endblock %}
{% block content %}
  <h1>Автори</h1>
  <table>
    <thead>
    <tr>
      <th>Ім'я</th>
      <th>Текстів</th>
      <th>Слів</th>
      <th>Символів</th>
      <th>Відстань до n-грамів бібліотеки</th>
    </tr>
    </thead>
    <tbody>
    {% for author in authors %}
      <tr>
        <th><p>
          <a href="/author/{{ author.last_name }} {{ author.first_name }}">{{
author.last_name }} {{ author.first_name }}</a>
          {% if author.date_birth %}
            ({{ author.date_birth | year }}{% if author.date_death %}&ndash;{{
author.date_death | year }}{% endif %})
          {% endif %}
        </p></th>
```

```
            <td><b>{{ author.n_texts }}</b></td>
            <td><b>{{ '{:,}'.format(author.n_words).replace(',', ' ') }}</b></td>
            <td><b>{{ '{:,}'.format(author.n_letters).replace(',', ' ') }}</b></td>
            <td>
               {% for distance in author.distance_to_library %}
                  {{ loop.index }}: <b>{{ '%0.3f'|format(distance) }}</b>  
               {% endfor %}
            </td>
         </tr>
      {% endfor %}
      </tbody>
   </table>
{% endblock %}
```

./app/templates/base.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <title>{% block title %}{% endblock %}</title>
   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
<header>
   <ul class="navbar">
      <li class="{% block menu_active_library %}{% endblock %}"><a href="{{
url_for('library_page') }}">Бібліотека</a></li>
      <li class="{% block menu_active_authors %}{% endblock %}"><a href="{{
url_for('authors_page') }}">Автори</a></li>
      <li class="{% block menu_active_reports %}{% endblock %}"><a href="{{
url_for('reports_page') }}">Звіти</a></li>
      <li class="{% block menu_active_analyze %}{% endblock %}"><a href="{{
url_for('analyze_page') }}">Аналізувати текст</a></li>
   </ul>
</header>
{% with messages = get_flashed_messages() %}
```

```
    {% if messages %}
        <ul class="messages">
            {% for message in messages %}
                <li>{{ message }}
                    <button onclick="this.parentElement.remove()">Закрити</button>
                </li>
            {% endfor %}
        </ul>
    {% endif %}
{% endwith %}
{% block content %}{% endblock %}
</body>
</html>
```

./app/templates/library.html

```
{% extends "base.html" %}
{% block title %}{{ library.title }}{% endblock %}
{% block menu_active_library %}active{% endblock %}
{% block content %}
    <h1>{{ library.title }}</h1>
    <table>
        <tr>
            <td><a href="{{ url_for('static', filename='authors/library/images/gram.png') }}"><img
src="{{ url_for('static', filename='authors/library/images/gram.png') }}" alt="Грам"
width="100%"></a></td>
            <td><a href="{{ url_for('static', filename='authors/library/images/bigram.png') }}"><img
src="{{ url_for('static', filename='authors/library/images/bigram.png') }}" alt="Біграм"
width="100%"></a></td>
        </tr>
    </table>
    <h3>Параметри бібліотеки</h3>
    <table>
        <thead>
        <tr>
            <th>Авторів</th>
            <th>Текстів</th>
```

```
            <th>Слів</th>
            <th>Символів</th>
        </tr>
        </thead>
        <tbody>
        <tr>
            <th><p>{{ library.n_authors }}</p></th>
            <td><b>{{ library.n_texts }}</b></td>
            <td><b>{{ '{:,}'.format(library.n_words).replace(',', ' ') }}</b></td>
            <td><b>{{ '{:,}'.format(library.n_letters).replace(',', ' ') }}</b></td>
        </tr>
        </tbody>
    </table>
    <h3>Слова, присутні у кожного з авторів ({{ library.common_words | length }})</h3>
    {% for word in library.common_words[:-1] %}
        {{ word }},
    {% endfor %}
    {{ library.common_words[-1] }}.
{% endblock %}
```

. /app/templates/report.html

```
{% extends "base.html" %}
{% block title %}Аналіз тексту{% endblock %}
{% block content %}
    {% if report %}
        <h1>Аналіз тексту</h1>
        {% if report.distance_to_authors %}
            <table>
                <tr>
                    <td><a href="/static/reports/{{ report_id }}/gram.png">
                        <img src="/static/reports/{{ report_id }}/gram.png" alt="Грам"
width="100%"></a>
                    </td>
                    <td><a href="/static/reports/{{ report_id }}/bigram.png">
                        <img src="/static/reports/{{ report_id }}/bigram.png" alt="Біграм"
width="100%"></a>
```

```
        </td>
      </tr>
    </table>
      <h3>Параметри тексту</h3>
      <table>
        <thead>
        <tr>
          <th>Назва</th>
          <th>Слів</th>
          <th>Символів</th>
        </tr>
        </thead>
        <tbody>
        <tr>
          <td><b>{{ report.title }}</b></td>
          <td><b>{{ '{:,}'.format(report.n_words).replace(',', ' ') }}</b></td>
          <td><b>{{ '{:,}'.format(report.n_letters).replace(',', ' ') }}</b></td>
        </tr>
        </tbody>
      </table>
    <h3>Найближчі автори</h3>
    <table class="distances">
      <tr>
        {% for distance in report.distance_to_authors %}
          <td>{{ loop.index }}-грам</td>
        {% endfor %}
      </tr>
      <tr>
        {% for distance in report.distance_to_authors %}
          <td>{{ '%0.3f'|format(distance[1]) }}<br>
            <a href="/author/{{ distance[0] }}"><b>{{ distance[0] }}</b></a>
          </td>
        {% endfor %}
      </tr>
    </table>
    <h3>Текст</h3>
```

```
        <label for="text"></label><textarea id="text" rows="40" style="width: 100%">{{
report.text | safe }}</textarea>
        {% else %}
        <h2>Відбувається створення звіту, поновіть сторінку за хвилину...</h2>
      {% endif %}
    {% else %}
      <h2>Звіт не знайдено</h2>
    {% endif %}
{% endblock %}
```

. /app/templates/reports.html

```
{% extends "base.html" %}
{% block title %}Звіти{% endblock %}
{% block menu_active_reports %}active{% endblock %}
{% block content %}
  <h1>Звіти</h1>
  <table>
    <thead>
    <tr>
      <th>Номер</th>
      <th>Назва</th>
      <th>Слів</th>
      <th>Символів</th>
    </tr>
    </thead>
    <tbody>
    {% for report in reports %}
      <tr>
        <th><b>{{ report.id }}</b></th>
        <td><p><a href="/report/{{ report.id }}"><b>{{ report.title }}</b></a></p></td>
        <td><b>{{ '{:,}'.format(report.n_words).replace(',', ' ') }}</b></td>
        <td><b>{{ '{:,}'.format(report.n_letters).replace(',', ' ') }}</b></td>
      </tr>
    {% endfor %}
    </tbody>
  </table>
```

```
{% endblock %}
```

./app/static/css/style.css

```
body {
    max-width: 1000px;
    margin: auto;
    border: 1px solid grey;
    padding: 20px;
    padding-top: 0;
    background-color: #ffffff;
}
ul.navbar {
    list-style-type: none;
    margin: 0 -20px;
    padding: 0;
    overflow: hidden;
    background-color: #555555;
}
.navbar li {
    float: left;
}
.navbar li a {
    display: block;
    color: white;
    padding: 15px;
    text-decoration: none;
    font-weight: bold;
    font-family: sans-serif;
}
.navbar li a:hover,
.navbar li.active {
    background-color: #111;
}
table {
    width: 100%;
    background-color: #ffffff;
```

```
}
table, th, td {
    border: 1px solid grey;
    border-collapse: collapse;
}
table tr:nth-child(even) {
    background-color: #efefef;
}
table thead {
    background-color: #cccccc;
}
thead th {
    padding: 10px;
    text-align: left;
}
th, td {
    padding: 2px 10px;
    text-align: left;
}
table.no-border,
table.no-border>tbody>tr>td {
    border: 0;
    padding: 0;
}
.distances td {
    padding: 10px;
}
.error {
    color: red;
    font-style: italic;
    font-size: small;
}
ul.messages {
    list-style-type: none;
    padding: 0;
}
```

```
ul.messages li {
    padding: 10px;
    background-color: #efefef;
}
```

./requirements.txt

```
alembic==1.4.2
beautifulsoup4==4.9.1
certifi==2020.4.5.1
chardet==3.0.4
click==7.1.2
cycler==0.10.0
Flask==1.1.2
Flask-Migrate==2.5.3
Flask-SQLAlchemy==2.4.3
Flask-WTF==0.14.3
html2text==2020.1.16
idna==2.9
itsdangerous==1.1.0
Jinja2==2.11.2
kiwisolver==1.2.0
Mako==1.1.2
MarkupSafe==1.1.1
matplotlib==3.2.1
numpy==1.18.4
psycopg2-binary==2.8.5
pyparsing==2.4.7
python-dateutil==2.8.1
python-dotenv==0.13.0
python-editor==1.0.4
PyYAML==5.3.1
requests==2.23.0
six==1.15.0
soupsieve==2.0.1
SQLAlchemy==1.3.17
urllib3==1.25.9
```

Werkzeug==1.0.1

WTForms==2.3.1

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО

Керівник проєкту

_____*Олексій ФІНОГЕНОВ*

(підпис) (вл. ім’я, прізвище)

“13” квітня 2020 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *Олександр ПАВЛОВ*

(підпис) (вл. ім’я, прізвище)

“14” квітня 2020 р.

Система визначення параметрів автора
за текстами творів

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6111.01.000 ТЗ*

на 9 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ…….. 3

1.1 Повне найменування системи та її умовне позначення…….. 3

1.2 Найменування організації-замовника та організацій-учасників робіт…… 3

1.3 Перелік документів, на підставі яких створюється система (Завдання на ДП)……... 3

1.4 Планові терміни початку і закінчення роботи зі створення системи……... 3

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ ……..... 4

2.1 Призначення системи…….. 4

2.2 Цілі створення системи…….. 4

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ…… 5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……..... 6

4.1 Вимоги до функціональних характеристик…….. 6

4.2 Вимоги до надійності…….. 6

4.3 Умови експлуатації (тільки для систем, специфіка яких передбачає особливі умови експлуатації)……. 6

4.4 Вимоги до складу і параметрів технічних засобів…… 6

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ……. 8

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ……. 9

6.1 Види випробувань…… 9

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: «Система визначення параметрів автора за текстами творів».

Коротке найменування системи: «Text Analyzer».

1.2 Найменування організації-замовника та організації-учасника робіт

Замовником є кафедра автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» (далі за текстом — Замовник). Адреса замовника: м. Київ, п. Перемоги 37, 18 корпус ФІОТ.

Розробник сервісу — студент групи ІС-з61 кафедри автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» Мак Олексій Володимирович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

– ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;

– ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;

– ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплектність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням системи – 13 квітня 2020 року.

Плановий термін по закінченню роботи над системою – 2 червня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Система визначення параметрів автора за текстами творів призначена для проведення досліджень з атрибуції авторства текстів українських письменників, авторство яких невідомо.

Окрім атрибуції авторства, система, надає можливість визначення параметрів можливого автора, таких як епоха творчості, географічне розташування, тощо. А автоматизований аналіз даних дозволить зберегти час, який користувач повинен був витратити на статистичний аналіз текстів та розрахунки.

2.2 Цілі та задачі створення системи

Цілі створення системи атрибуції авторства текстів:

- створення корпусу прозових літературних творів українських письменників;
- побудова, на основі створеного корпусу еталонних стилеметричних портретів авторів, включених до корпусу;
- проведення аналізу та атрибуції авторства текстів, за допомогою обрахованих статистичних портретів;
- автоматизація процесу проведення досліджень.

Отже, головною метою розробки є створення інструменту для визначення авторства текстів за допомогою статистичних методів.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об'єктом автоматизації є процес визначення параметрів автора літературного твору, авторство якого не є відомим.

Отже, розробка має надавати адміністраторам системи таку функціональність:

- Додати авторів до бібліотеки,
- Додати тексти авторів,
- Додати інформацію про авторів,
- Змінити налаштування системи.

В той же час користувачі системи повинні мати можливість:

- Переглянути параметри бібліотеки,
- Переглянути перелік авторів,
- Переглянути параметри автора,
- Переглянути тексти автора,
- Проаналізувати текст,
- Переглянути перелік звітів,
- Переглянути звіт.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

- Система має бути розрахована на виконання великих об’ємів обчислень за допомогою паралельних обчислень та масштабувати виконання операцій на усі, наявні на сервері процесорні ядра.
- Архітектура системи повинна мати клієнт-серверну архітектуру та мати змогу бути встановленою як на локальний комп’ютер дослідника, так і на потужний хмарний сервер.
- Система має бути придатною для розширення функціоналу.
- Система має надавати змогу зберігати, завантажувати та видаляти інформацію про наявних у ній авторів та проведені дослідження.

4.2 Вимоги до надійності

Система повинна забезпечувати надійність одразу на декількох рівнях:

- На рівні програмного коду, опрацьовувати усі можливі виключні ситуації, та запобігати аварійному завершенню роботи.
- На рівні системи зберігання даних – використовувати стандартизовані системи керування базами даних та файлових сховищ.

4.3 Умови експлуатації

Усі користувачі системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

У випадку встановлення системи на віддалений сервер, користувачі та адміністратор мають дотримуватись правил інформаційної безпеки.

4.4 Вимоги до складу та параметрів технічних засобів

Поставлена перед системою задача, а саме порівняння величезних векторів є справді дуже ресурсозатратною, з точки зору обчислювальних

можливостей. Саме тому, для створення дата сету, обробки даних та проведення подальших досліджень було використано сервер, оснащений 8-ма ядрами та 16 гігабайтами оперативної пам’яті.

Хоча мінімальні вимоги до технічного забезпечення серверу для запуску системи є низькими:

- кількість ядер процесора – 1,
- об’єм оперативної пам’яті – 1 Гб,
- об’єм дискового накопичувача – 10 Гб,
- мережевий інтерфейс,
- операційна система сімейства Linux (CentOS, Debian, Ubuntu).

Для використання системи з великим дата сетом авторів та проведення масових аналізів невідомих текстів, система має задовільняти рекомендованій конфігурації:

- кількість ядер процесора – 8,
- об’єм оперативної пам’яті – 12 Гб,
- об’єм дискового накопичувача – 32 Гб,
- мережевий інтерфейс,
- операційна система сімейства Linux (CentOS, Debian, Ubuntu).

Незалежно від конфігурації, на сервері має бути встановлено середовище для виконання програм Python 3.8.

Вимоги до клієнтського технічного забезпечення є мінімальними, для роботи з системою потрібен будь-який сучасний веббраузер та мережеве підключення.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ з/п	Назва етапів створення продукту	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	13.04.2020	Технічне завдання підготовлено та узгоджено з замовником
2.	Розробка сценарію роботи	17.04.2020	Сценарій роботи розроблено
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	23.04.2020	Цілі, задачі та функціональність визначені, технічне проектування завершено
4.	Узгодження з керівником інтерфейсу користувача	30.05.2020	Інтефейс користувача узгоджено з керіником
5.	Алгоритмізація задачі	01.05.2020	Розроблено основний алгоритм роботи застосунку
6.	Розробка інформаційного забезпечення	05.05.2020	Розробка інформаційного забезпечення завершена
7.	Розробка програмного забезпечення	19.05.2020	Розробка програмного забезпечення завершена
8.	Налагодження програми	24.05.2020	Програму налагоджено
9.	Тестування програми	25.05.2020	Тестування завершено, застосунок успішно пройшов усі випробування
10.	Здача готового програмного продукту замовнику	30.05.2020	Готовий програмний продукт сдано замовнику

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

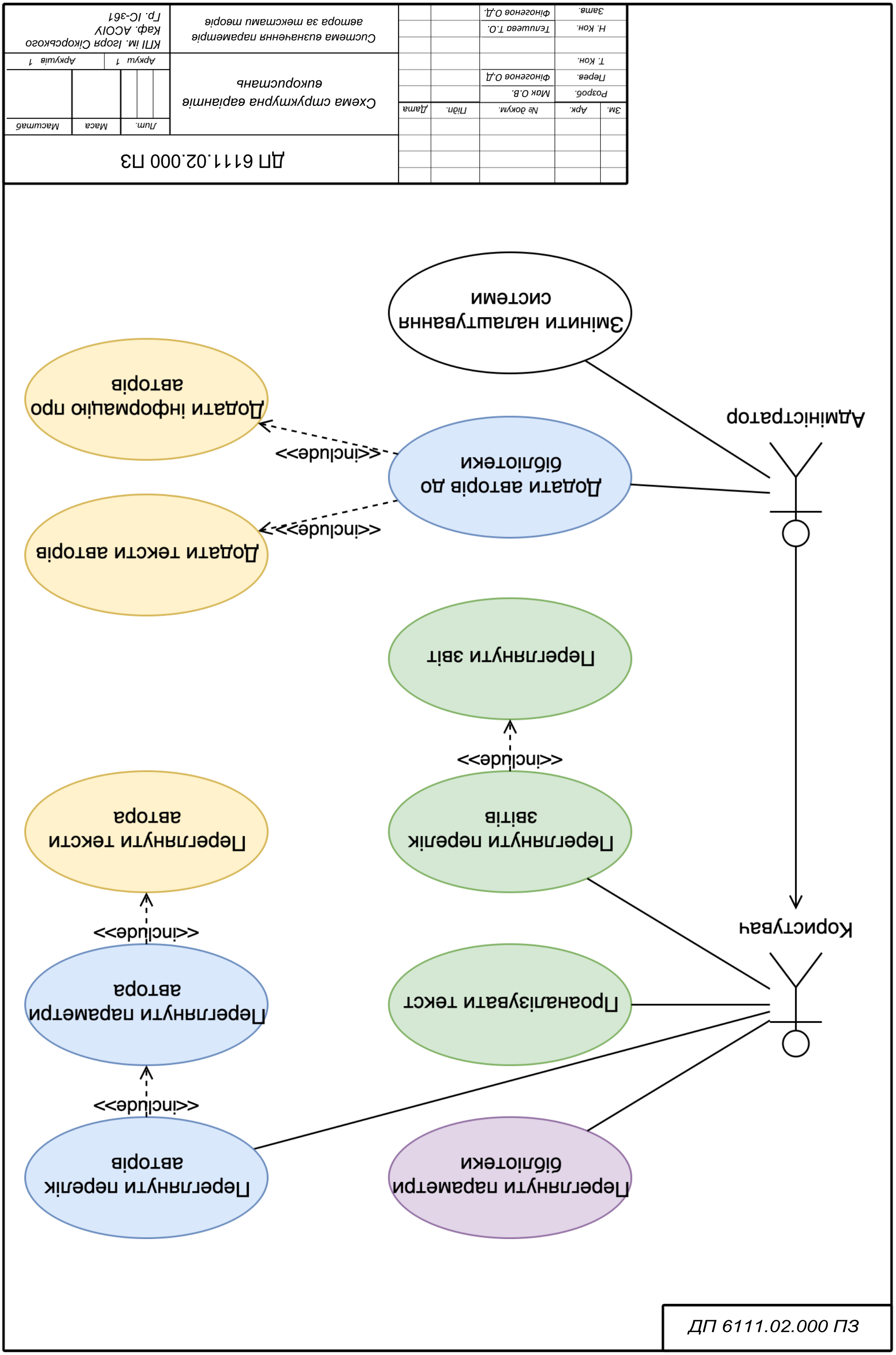
Види випробувань узгоджуються із замовником до проведення випробувань. Здача – прийом робіт виконується поетапно на комп'ютерах замовника в аудиторіях кафедри АСОІУ у відповідності з робочою програмою та календарним планом.

Усі програмні продукти, що створюються в рамках даної системи передаються замовнику як у вигляді готових модулів, так і у вигляді вихідних кодів, представлених в електронній формі.

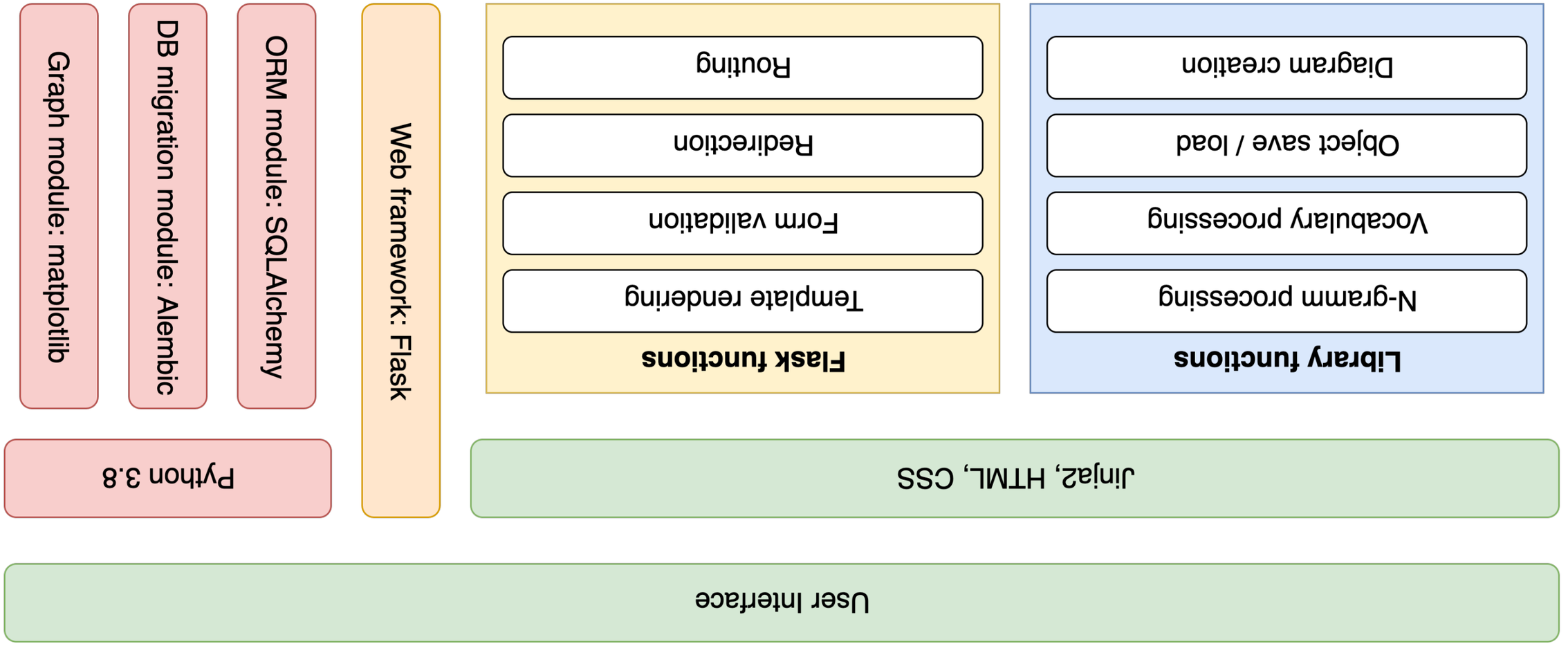
Графічний матеріал
до дипломного проєкту

на тему: *«Система визначення параметрів автора за текстами творів»*

Київ – 2020 року



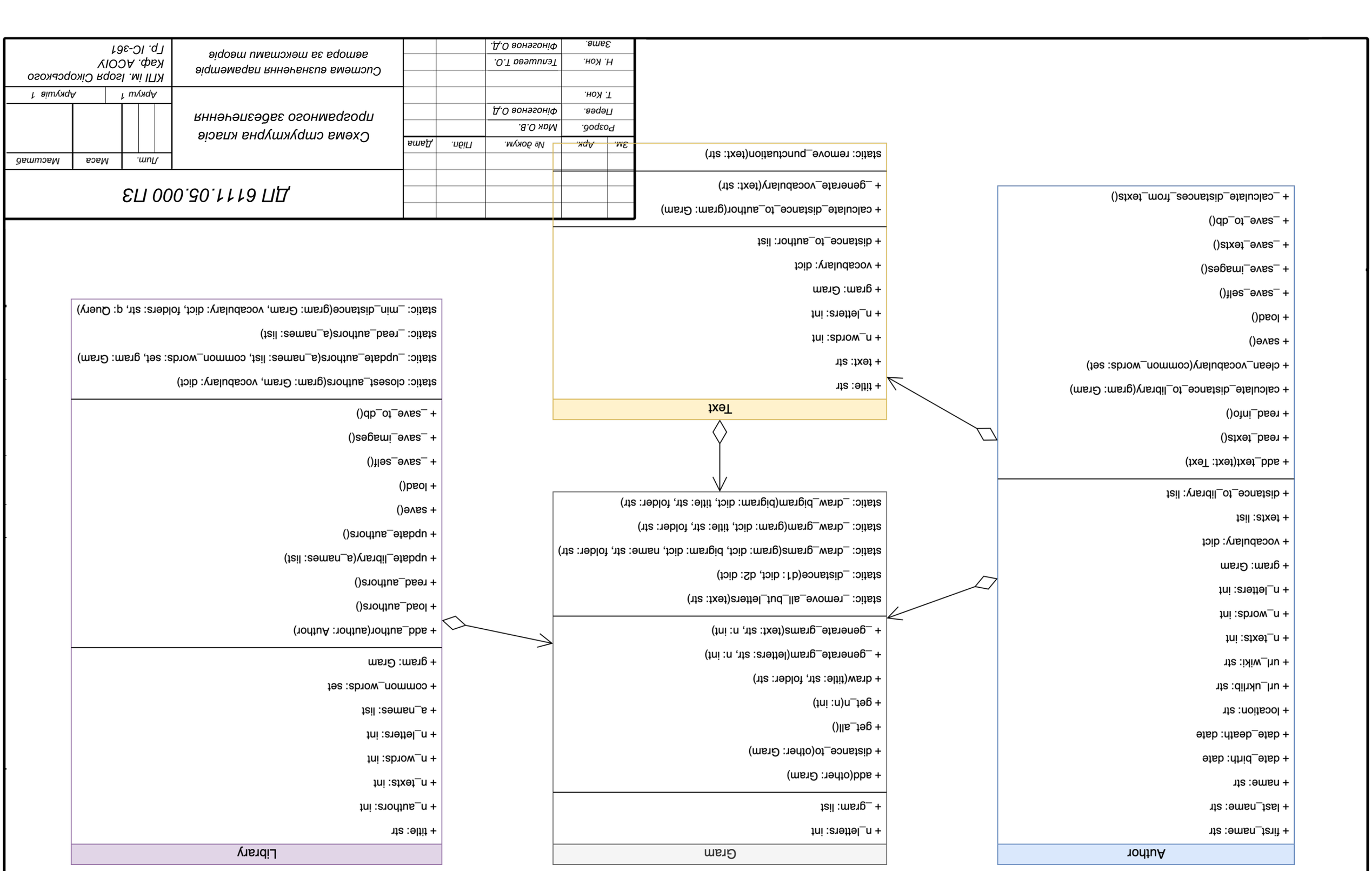
								ДП 6111.02.000 ПЗ	
Змн.	Арк.	№ докум.	Підп.	Дата					
Розроб.		Мак О.В.			Схема структура взаємодії користувачів			Літ.	Маса	Масштаб
Перев.		Фіногенов О.Д.								
Т. кон.								Арк. 1	Аркушів 1	
Н. Кон.		Тепцеев Т.О.								
Затв.		Фіногенов О.Д.			Система взаємодії параметрів автора за мекстами теорія			КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-361		



					ДП 6111.03.000 ПЗ
Зм.	Арк.	№ докум.	Підп.	Дата	
Розроб.		Мах О.В.			Кредитний відділ екранних форм
Перев.		Фіногенов О.Д.			
Т. Конс.					
Н. Конт.		Тепцеєв Т.О.			Система визначення параметрів вектора за текстами моделі
Затв.		Фіногенов О.Д.			
Літ.		Маса	Масштаб		КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-361
Арк. 1		Аркушів 1			

ДП 6111.04.000 ПЗ

Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6111.04.000 ПЗ		
Розроб.		Мак О.В.			Схема бази даних	Літ.	Маса	Масштаб
Перев.		Фіногенов О.Д.						
Т. Кон.						Аркуш 1	Аркушів 1
Н. Кон.		Телцевед Т.О.			Система вцзначення параметрів автора за мекстами авторів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-361	
Затв.		Фіногенов О.Д.					



ДП 6111.05.000 ПЗ

Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.	Мак О.В.			
Перев.	Фіногенов О.Д.			
Т. Кон.				
Н. Кон.	Тепшеєв Т.О.			
Затв.	Фіногенов О.Д.			

ДП 6111.05.000 ПЗ

Схема структура класів програмного забезпечення

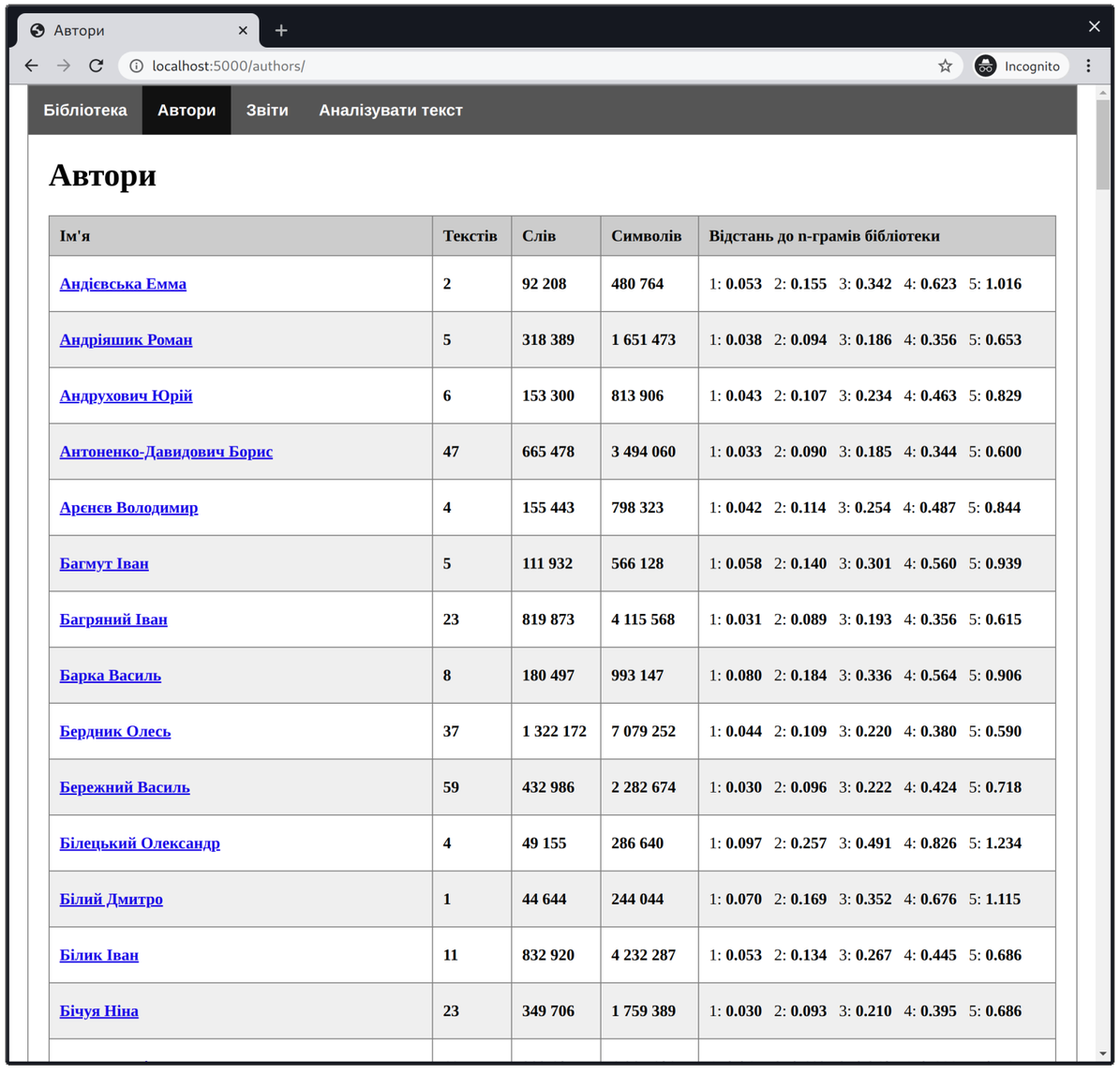
Літ.	Маса	Масштаб

Аркуш 1 | Аркушів 1

КПІ ім. Ігоря Сікорського
Кафедра АСОІУ
Гр. ІС-361

Система вилучення параметрів автора за текстами творів

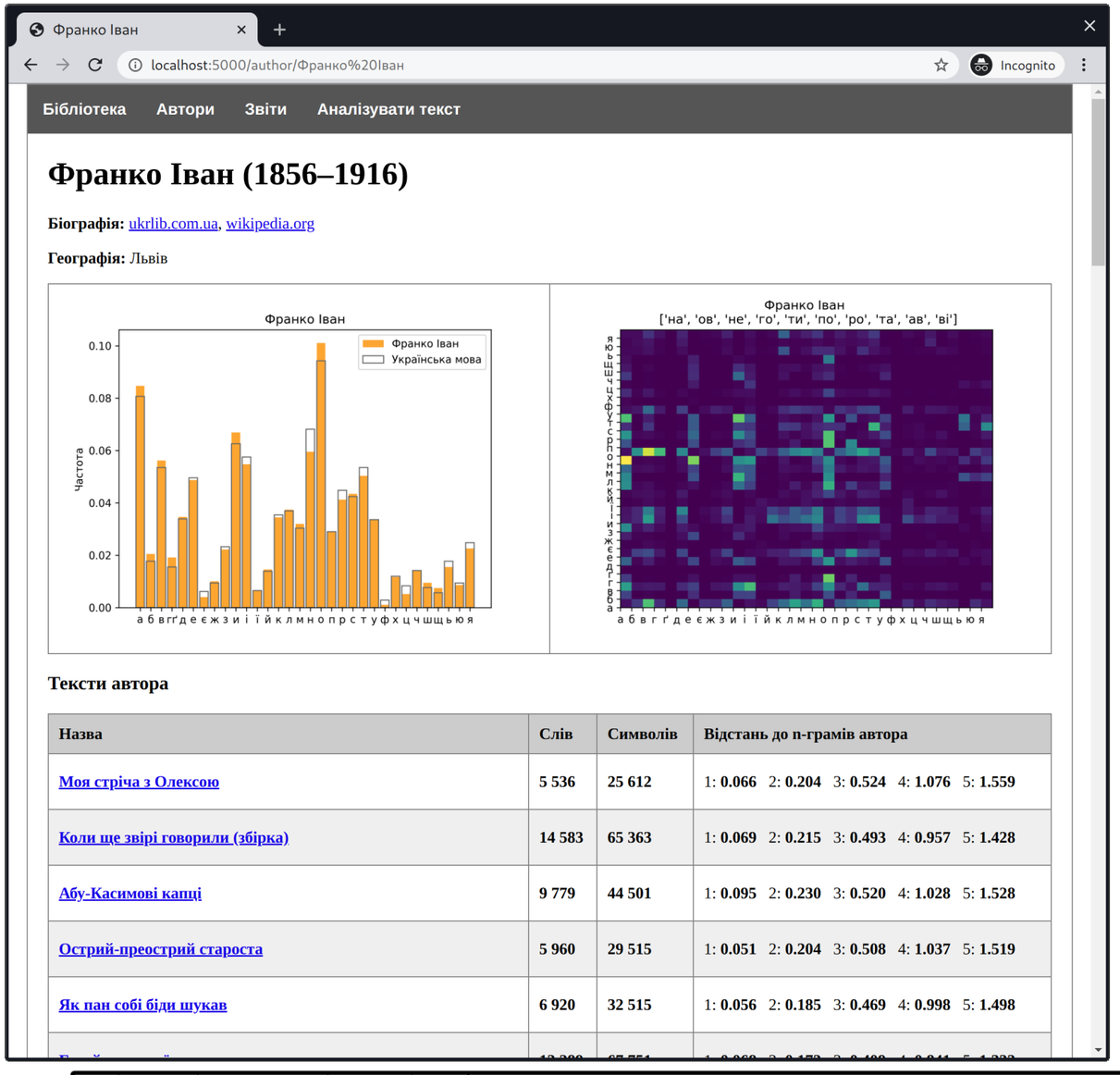
Екранна форма головної сторінки розділу Автори



Автори

Ім'я	Текстів	Слів	Символів	Відстань до n-грамів бібліотеки
Андієвська Емма	2	92 208	480 764	1: **0.053** 2: **0.155** 3: **0.342** 4: **0.623** 5: **1.016**
Андріяшик Роман	5	318 389	1 651 473	1: **0.038** 2: **0.094** 3: **0.186** 4: **0.356** 5: **0.653**
Андрухович Юрій	6	153 300	813 906	1: **0.043** 2: **0.107** 3: **0.234** 4: **0.463** 5: **0.829**
Антоненко-Давидович Борис	47	665 478	3 494 060	1: **0.033** 2: **0.090** 3: **0.185** 4: **0.344** 5: **0.600**
Арєнєв Володимир	4	155 443	798 323	1: **0.042** 2: **0.114** 3: **0.254** 4: **0.487** 5: **0.844**
Багмут Іван	5	111 932	566 128	1: **0.058** 2: **0.140** 3: **0.301** 4: **0.560** 5: **0.939**
Багряний Іван	23	819 873	4 115 568	1: **0.031** 2: **0.089** 3: **0.193** 4: **0.356** 5: **0.615**
Барка Василь	8	180 497	993 147	1: **0.080** 2: **0.184** 3: **0.336** 4: **0.564** 5: **0.906**
Бердник Олесь	37	1 322 172	7 079 252	1: **0.044** 2: **0.109** 3: **0.220** 4: **0.380** 5: **0.590**
Бережний Василь	59	432 986	2 282 674	1: **0.030** 2: **0.096** 3: **0.222** 4: **0.424** 5: **0.718**
Білецький Олександр	4	49 155	286 640	1: **0.097** 2: **0.257** 3: **0.491** 4: **0.826** 5: **1.234**
Білий Дмитро	1	44 644	244 044	1: **0.070** 2: **0.169** 3: **0.352** 4: **0.676** 5: **1.115**
Білик Іван	11	832 920	4 232 287	1: **0.053** 2: **0.134** 3: **0.267** 4: **0.445** 5: **0.686**
Бічуя Ніна	23	349 706	1 759 389	1: **0.030** 2: **0.093** 3: **0.210** 4: **0.395** 5: **0.686**

Екранна форма сторінки автора



Франко Іван (1856–1916)

Біографія: ukrlib.com.ua, wikipedia.org

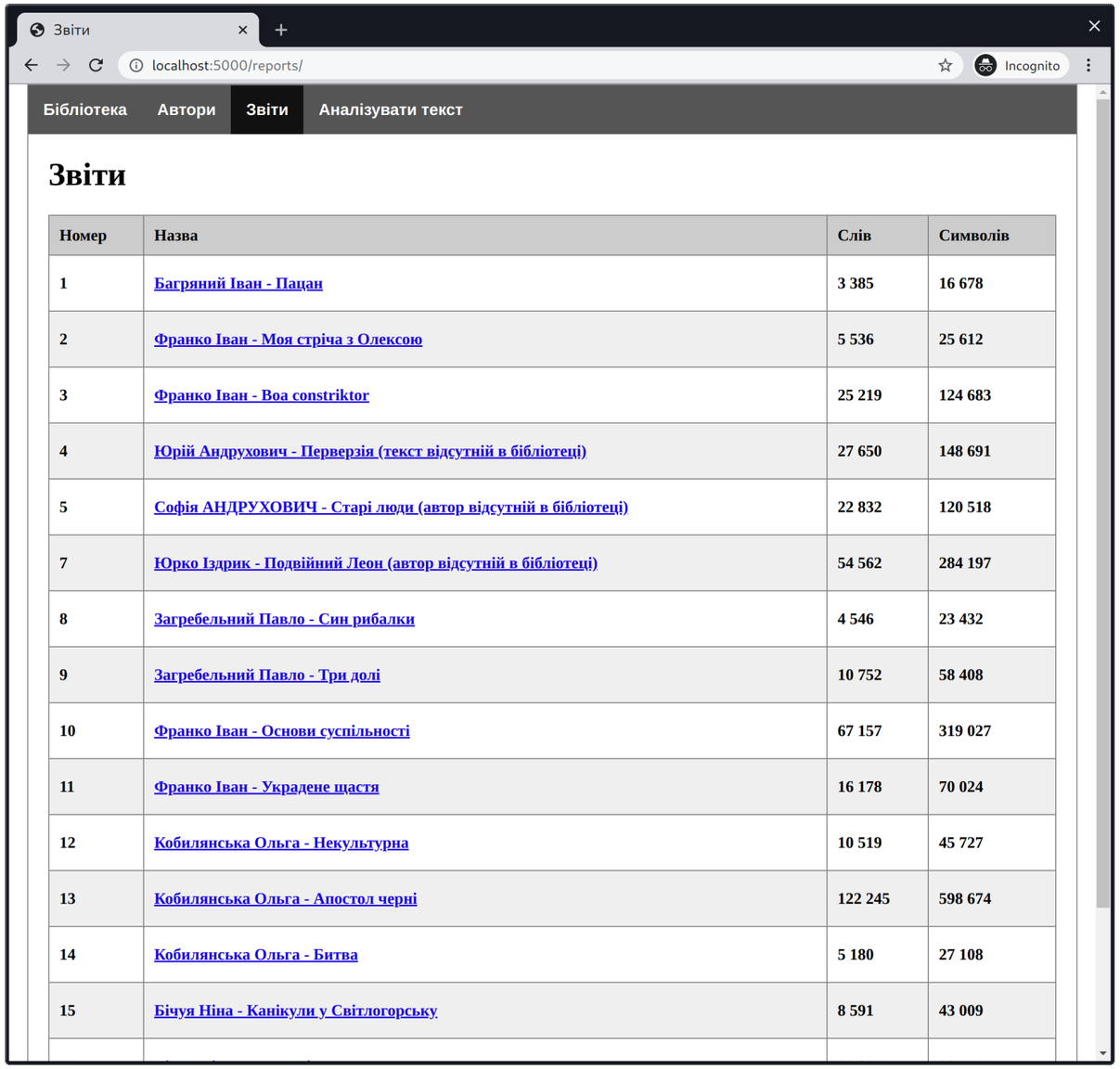
Географія: Львів

Тексти автора

Назва	Слів	Символів	Відстань до n-грамів автора
Моя стріча з Олексою	5 536	25 612	1: **0.066** 2: **0.204** 3: **0.524** 4: **1.076** 5: **1.559**
Коли ще звірі говорили (збірка)	14 583	65 363	1: **0.069** 2: **0.215** 3: **0.493** 4: **0.957** 5: **1.428**
Абу-Касимові капці	9 779	44 501	1: **0.095** 2: **0.230** 3: **0.520** 4: **1.028** 5: **1.528**
Острий-преострий староста	5 960	29 515	1: **0.051** 2: **0.204** 3: **0.508** 4: **1.037** 5: **1.519**
Як пан собі біди шукав	6 920	32 515	1: **0.056** 2: **0.185** 3: **0.469** 4: **0.998** 5: **1.498**

Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6111.06.000 КЕ		
Розроб.		Мак О.В.			Креслення вигляду екранних форм	Лит.	Маса / Масштаб
Перев.		Фіногенов О.Д				Аркуш 1	Аркушів 3
Т. Кон.					Система визначення параметрів автора за текстами творів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-з61	
Н. Кон.		Тєлишева Т.О.					
Затв.		Фіногенов О.Д.					

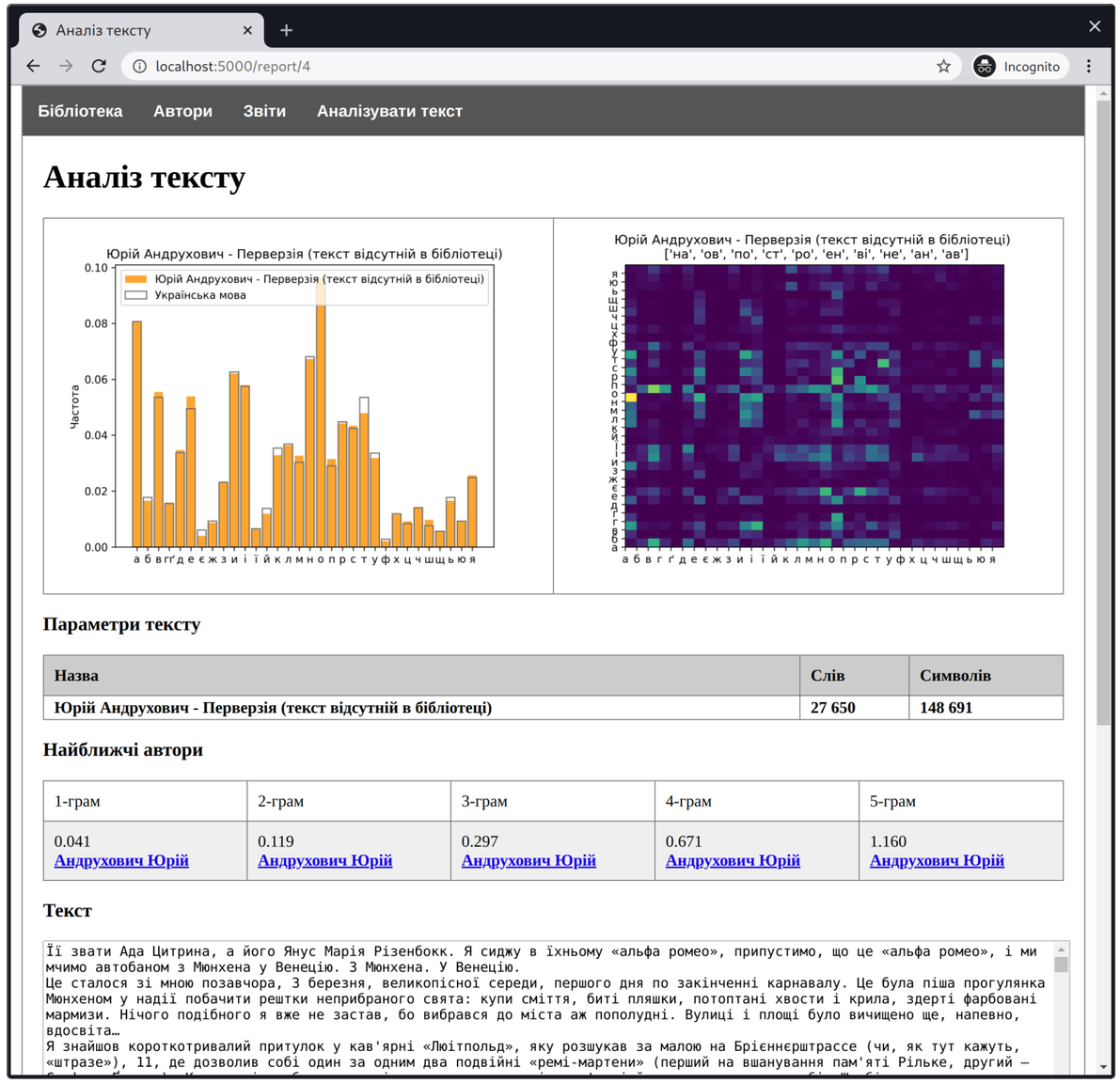
Екранна форма головної сторінки розділу Звіти



Звіти

Номер	Назва	Слів	Символів
1	Багряний Іван - Пацан	3 385	16 678
2	Франко Іван - Моя стріча з Олексою	5 536	25 612
3	Франко Іван - Boa constriktor	25 219	124 683
4	Юрій Андрухович - Перверзія (текст відсутній в бібліотеці)	27 650	148 691
5	Софія АНДРУХОВИЧ - Старі люди (автор відсутній в бібліотеці)	22 832	120 518
7	Юрко Ізdрик - Подвійний Леон (автор відсутній в бібліотеці)	54 562	284 197
8	Загребельний Павло - Син рибалки	4 546	23 432
9	Загребельний Павло - Три долі	10 752	58 408
10	Франко Іван - Основи суспільності	67 157	319 027
11	Франко Іван - Украдене щастя	16 178	70 024
12	Кобилянська Ольга - Некультурна	10 519	45 727
13	Кобилянська Ольга - Апостол черні	122 245	598 674
14	Кобилянська Ольга - Битва	5 180	27 108
15	Бічуя Ніна - Канікули у Світлогорську	8 591	43 009

Екранна форма сторінки звіту



Аналіз тексту

Параметри тексту

Назва	Слів	Символів
Юрій Андрухович - Перверзія (текст відсутній в бібліотеці)	27 650	148 691

Найближчі автори

1-грам	2-грам	3-грам	4-грам	5-грам
0.041 Андрухович Юрій	0.119 Андрухович Юрій	0.297 Андрухович Юрій	0.671 Андрухович Юрій	1.160 Андрухович Юрій

Текст

Її звати Ада Цитрина, а його Янус Марія Різенбокк. Я сиджу в їхньому «альфа ромео», припустимо, що це «альфа ромео», і ми мчимо автобаном з Мюнхена у Венецію. З Мюнхена. У Венецію.
Це сталося зі мною позавчора, 3 березня, великопісної середи, першого дня по закінченні карнавалу. Це була піша прогулянка Мюнхеном у надії побачити рештки неприбраного свята: купи сміття, биті пляшки, потоптані хвости і крила, здерті фарбовані мармизи. Нічого подібного я вже не застав, бо вибрався до міста аж пополудні. Вулиці і площі було вичищено ще, напевно, вдосвіта…
Я знайшов короткотривалий притулок у кав'ярні «Люітпольд», яку розшукав за малою на Брієннєрштрассе (чи, як тут кажуть, «штразе»), 11, де дозволив собі один за одним два подвійні «ремі-мартени» (перший на вшанування пам'яті Рільке, другий —

Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6111.06.000 КЕ		
Розроб.		Мак О.В.			Креслення вигляду екранних форм	Лит.	Маса / Масштаб
Перев.		Фіногенов О.Д				Аркуш 2	Аркушів 3
Т. Кон.					Система визначення параметрів автора за текстами творів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-з61	
Н. Кон.		Телишева Т.О.					
Затв.		Фіногенов О.Д.					

Бібліотека × + localhost:5000/library/ Incognito

Бібліотека Автори Звіти Аналізувати текст

Бібліотека

Параметри бібліотеки

Автори	Текстів	Слів	Символів
181	2029	59 921 176	300 856 755

Слова, присутні у кожного з авторів (272)

а, або, але, б, багато, вагу, без, би, бо, боку, вдати, був, буде, бути, бґрунтувати, в, вас, великий, вже, взяв, ви, видно, вийшла, виходити, вона, воно, все, всіх, всю, вся, всі, всіх, від, він, говорив, говорю, говорять, давній, дали, два, дев'ять, день, дні, до, добре, другий, дуже, думав, думки, же, жити, жінки, з, за, завдяки, зайняти, знає, знати, зі, й, йде, його, йому, йти, кажуть, казав, кого, коли, колись, коло, кому, куди, кінець, легко, людей, мав, мало, матери, має, менше, мені, ми, много, може, моя, моє, мої, мій, міг, міг, місце, місця, місці, місяць, на, над, назад, нам, нас, наша, наші, не, нево, неї, ним, ними, них, новий, ночі, ночі, нього, ні, ній, ніж, ній, ніколи, ніхто, нічого, о, один, одна, одного, одному, одну, от, очей, очима, очі, перед, перший, по, потім, при, про, просто, під, після, пішли, пішов, раз, разом, робити, роботу, рук, руками, руках, руки, руки, рукою, руку, тік, сам, сама, саме, самого, самому, сами, свого, свою, своє, своєму, своєю, своєї, свої, своїй, своїм, своїми, своїх, свій, свій, себе, сере, середі, серце, сидів, сиділа, сказав, сказала, сказати, слова, слово, словом, слід, сніг, стало, стала, стало, стояв, та, так, така, таке, такий, такий, такими, таких, такого, такої, такі, там, те, тебе, тепер, ти, тим, тих, то, тобі, того, той, тому, трева, три, трохи, ту, тут, ті, тільки, у, уже, усе, усі, хотів, хоч, хоче, хто, хіба це, час, часу, через, чи, чим, чого, чомусь, чотири, чув, шукати, шастя, ще, що, щоб, щось, я, як, яка, яке, який, якийсь, яким, яких, якого, якої, якої, є, і, із, їй, їм, їх, її.

Екранна форма стопінки параметрів бібліотеки

Бібліотека × + localhost:5000/analyze Incognito

Бібліотека Автори Звіти Аналізувати текст

Аналізувати текст

Назва

Текст

Аналіз може зайняти декілька хвилин, дочекайтесь завантаження сторінки заздалегідь.

Аналізувати Очистити

Екранна форма розділу Аналізувати текст

Зм.	Арк.	№ докум.	Підп.	Дата	ДП 6111.06.000 КЕ		
Розроб.		Мак О.В.			Кресленння вузлових екранних форм	Літ.	Маса	Масштаб
Перев.		Фіногенова О.Д.						
Т. Кон.						Аркуш 3		Аркушів 3
Н. Кон.		Тепцева Т.О.			Система вузначення параметрів автора за текстами теорія	КПІ ім. Ігора Сікорського Каф. АСОІУ Гр. ІС-361		
Затв.		Фіногенова О.Д.						